NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER

DEFENSE COMMUNICATIONS AGENCY

COMPUTER SYSTEM MANUAL
CSM PSM 9A-67
VOLUME III, PART A
29 FEBRUARY 1972

THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK)

SIMULATION OUTPUT SUBSYSTEMS

Vol. II. Pt F
AD742789

PROGRAMMING SPECIFICATIONS MANUAL

369

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) National Military Command System Support Center (NMCSSC) Defense Communications Agency (DCA) The Pentagon Washington, DC 20301	2a. REPORT SECURITY CLASSIFICATION 2b. GROUP

3. REPORT TITLE

The NMCSSC Quick-Reacting General War Gaming System (QUICK) Programming Specifications Manual, Volume III, Simulation (Output) Subsystem

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

N/A

5. AUTHOR(S) (First name, middle initial, last name)

NMCSSC: Yvonne Mapily
Donald F. Webb

Lambda Corp: Lawrence B. Dean, Jr.
Jack A. Sasseen

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
29 February 1972	368	4

8a. CONTRACT OR GRANT NO. DCA 100-70-C-0065 b. PROJECT NO. NMCSSC Project 631 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) NMCSSC COMPUTER SYSTEM MANUAL CSM PSM 9A-67 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) None

10. DISTRIBUTION STATEMENT

This document is approved for public release; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	National Military Command System Support Center/Defense Commu.. ations Agency The Pentagon, Washington, DC 20301

13. ABSTRACT

This is one of three volumes describing computer programs of the QUICK-Reacting General War Gaming System (QUICK). These volumes complement other NMCSSC Computer System Manuals on QUICK by discussing the programs from a computer programming point of view. This volume, in two parts, concentrates on the Simulation (Output) Subsystem of QUICK. Other volumes are available for the Input Subsystem and Plan Generation Subsystem. Collectively, these manuals provide a basis for maintenance activity on the QUICK System.

Based upon a suitable data base, and user control parameters, QUICK will generate individual bomber and missile plans suitable for war gaming. The generated plans are of a form suitable for independent review and revision. Subsequently, execution of the planned events can be simulated. Various statistical summaries can be produced to reflect the results of the war game. A variety of force postures and strategies can be accommodated.

QUICK is documented extensively in a set of Computer System Manuals (series 9-67) published by the National Military Command System Support Center (NMCSSC), Defense Communications Agency (DCA), The Pentagon, Washington, DC 20301.

DD FORM 1 NOV 65 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

357

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT

Security Classification

NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER

Computer System Manual Number CSM PSM 9A-67

29 February 1972

THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK)

Programming Specifications Manual

Volume III - Simulation (Output) Subsystems

Part A

Submitted by:

DONALD F. WEBB
Major, USAF
Project Officer

REVIEWED BY:

R. E. HARSHBARGER
Technical Director
NMCSSC

APPROVED BY:

BRUCE MERRITT
Colonel, USA
Commander, NMCSSC

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.

ACKNOWLEDGMENT

This document was prepared under the direction of the Chief for Development and Analysis, NMCSSC, in response to a requirement of the Studies, Analysis and Gaming Agency (SAGA), Organization of the Joint Chiefs of Staff. Technical support was provided by Lambda Corporation under Contract Number DCA 100-70-C-0065.

CONTENTS

Part A

Chapter Page

Part B

ILLUSTRATIONS

TABLES

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, simulate the planned events, and provide statistical output summaries. QUICK has been programmed in FORTRAN for use on the NMCSSC CDC 3800 computer system.

The QUICK Programming Specifications Manual (PSM) consists of three volumes: Volume I, Data Input Subsystem; Volume II, Plan Generation Subsystem; Volume III, Simulation and Data Output Subsystems. The Programming Specifications Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume III, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs of the Simulation and Data Output subsystems. This volume consists of two parts: Part A provides a description of the programs/subroutines which make up the two subsystems; Part B contains the associated program listings. Companion documents are:

1. GENERAL DESCRIPTION
   Computer System Manual CSM GD 9A-67
   A nontechnical description for senior management personnel

2. ANALYTICAL MANUAL
   Computer System Manual CSM AM 9A-67 (three volumes)
   Provides a description of the system methodology for the non-programmer analysts

3. USER'S MANUAL
   Computer System Manual CSM UM 9-67
   Provides detailed instructions for applications of the system

4. OPERATOR'S MANUAL
   Computer System Manual CSM OM 9A-67
   Provides instructions and procedures for the computer operators

CHAPTER 1
INTRODUCTION

The QUICK system consists of four functional subsystems: the Data Input, Plan Generation, Simulation, and Data Output subsystems.* This third volume of the Programming Specifications Manual describes the QUICK Simulation subsystem, hereafter referred to as the Simulator, and the Data Output subsystem. The Simulator accepts basic game data prepared by the Data Input subsystem, plus one or more plans prepared by the Plan Generator. The Simulator then calculates the possible detailed activities and results if the plan(s) were implemented. The Simulator prepares a "history tape" which is processed by the Data Output subsystem to give various reports to assist the user in evaluating these activities and results.

The Simulator consists of a single program, SIMULATE. It simulates the activities of the missiles, bombers, and tankers in a general war, as planned by the Plan Generator. While the Simulator provides certain end-of-game summary information, the Data Output subsystem provides for printing standard and special summary reports on the events which occurred during the game.

The Data Output subsystem consists of three programs:

1. READSUM, which prepares
   a. Standard summary reports
   b. Four Actual Ground Zero (AGZ) tapes used as input to non-QUICK systems to obtain damage assessments
   c. Two "formatted" history tapes (prepared in attribute-value form) for use in program TABGEN
2. TABGEN, which prepares special summary reports (tables) from the formatted history tapes produced by program READSUM
3. HISTP, which prints out a detailed time-sequenced history of selected aspects of the game, primarily for diagnostic use.

---

*The QUICK subsystems are also referenced by the names Input subsystem, Plan Generator, Simulator, and Output subsystem.

The general concepts of operation and the analytical aspects of the design of the Simulator are presented in Volume III of the Analytical Manual. A detailed description of the user-input parameters required for operating the Simulation and Data Output subsystems is contained in the User's Manual, Volume II (see chapters 4 and 5 respectively).

Chapter 2 of this manual provides a detailed description of the Simulator, while chapters 3 to 5 describe the programs of the Data Output subsystem. Within each chapter, the initial sections describe the concept of operation and provide a description of the input/output files and common blocks associated with each program. Subsequent sections of each chapter describe the subroutines which constitute the program.

QUICK GENERAL-PURPOSE UTILITY PACKAGE

In addition to the main programs of the four QUICK subsystems, QUICK employs a general-purpose utility package. This utility package consists of programs, subroutines, and functions which perform a variety of support tasks common to two or more system programs. These programs and routines are discussed in chapters 2, 3, and 4 of the Programming Specifications Manual (PSM), Volume I, Data Input Subsystem and, where appropriate, in chapter 1 of the QUICK User's Manual, Volume II (see Special-Purpose Utility Routines). Appendix B of this volume contains a list of the entry points within the utility programs.

The QUICK System Filehandler

The QUICK system filehandler uses a word stream concept of operation. For the calling program, the filehandler retrieves or sends a stream of words from/to the input/output (I/O) devices. Thus, the programmer need never consider the makeup of the logical or physical records on the tape or disk for filehandler files. Only in the filehandler itself need the maintenance programmer be concerned with the physical characteristics of I/O. In the using program, input/output on filehandler files consists of a word stream. The program merely requests transfer of a number of words to/from the device. Thus, a description of the physical record structure of filehandler files is irrelevant to the maintenance programmer except when he is maintaining the filehandler subroutines themselves.

The programs of the Simulation and Data Output subsystems use the filehandler in conjunction with input/output operations. The filehandler subroutines and their functions are summarized below. A detailed description of the QUICK filehandler is contained in chapter 2 of the Programming Specifications Manual, Volume I.

SUBROUTINE	FUNCTION
ALOCDIR	Initializes disk file directory
INITAPE	Initializes filehandler
DEACTIV	Removes file name from active list
SETREAD	Prepares file for reading
RDWORD	Transfers one word from file input buffer to common /TWORD/
RDARRAY	Transfers block of words from file input buffer to user-specified core storage area
SETWRITE	Prepares file for writing
WRWORD	Transfers one word from common /TWORD/ to file output buffer
WRARRAY	Transfers block of words from user-specified core storage area to file output buffer
TERMTAPE	Terminates files after reading or writing and releases buffer area for use by other files

COMPUTER STORAGE REQUIREMENTS

The NMCSSC CDC 3800 computer provides a maximum of 65,534 words of core storage. Excluding the requirements of the operating system, the core storage requirements of the programs of the Simulation and Data Ouput subsystems are as follows:

SIMULATE	55,305
READSUM	48,310
TABGEN	31,700
HISTP	13,820

CHAPTER 2
THE SIMULATION SUBSYSTEM

PURPOSE

SIMULATE performs a detailed war game of the plans generated by the QUICK Plan Generator. The plans are forwarded to the Simulator via one or more event tapes (EVENTAPEs) prepared by program PLNTPLAN. A plan consists of a series of desired occurrences (events) for a vehicle, which together make up the vehicle's planned mission (referred to as a sortie). The outcome of each event may be probabilistic or may depend on conditions existing at the time. The Simulator checks the conditions, and determines probabilistic events randomly, with each event occurring at its proper game time.

The events which can occur for a bomber are:

- Launch
- Refuel
- Enter Air Defense Zone
- Change Altitude
- Decoy Launch
- ASM Launch
- Area Attrition
- Local Attrition
- Abort
- Recovery

The events which can occur for a tanker are:

- Launch
- Enter Refuel Area
- Leave Refuel Area
- Abort
- Recovery

The events which can occur for a ballistic missile are:

Launch

Complete Launch (disperse MIRVs)

Area Ballistic Missile Defense

Terminal Ballistic Missile Defense

A damage subroutine is called whenever a bomber, ASM, or ballistic missile successfully delivers a warhead to a target.

INPUT FILES

Two files are input to program SIMULATE: the EVENTAPE and the SIMTAPE. The EVENTAPE, of which there is normally one for each side, is an output of program PLNTPLAN of the Plan Generation subsystem. A sortie for each vehicle (bomber, tanker, or missile) in the game is read into the array INDATA in common block /EDATA/, from where it is put in the list of events to be processed. The vehicle class is recognized from the value of the attribute* ICLASS, contained in the 10th word of the vehicle record. The EVENTAPE may also contain events for simulating time-dependent destruction-before-launch (DBL) of naval forces. Such records have 5HNAVAL in word 7. The EVENTAPE also contains tables of recovery base data for the appropriate side.

The SIMTAPE is an output of program INDEXER of the Data Input subsystem. It contains basic tables of characteristics of game elements. SIMULATE uses these tables to fill, at least in part, the following common blocks:

/AREADAT/

/ASMS/

/BOMBER/

/BRKPNT/

/CAPACITY/

/DAMAGE/

/MISSLE/

/NAMES/

*For attribute description, see appendix A.

/NAVDATA/

/NCOL/

/PAYLOAD/

/TANKER/

/TI [illegible] /

/WA[illegible]/

/ZONES/

/19501/

OUTPUT FILES

The only file output by SIMULATE is the HISTAPE, which contains a record of each occurrence or event in the game.

The HISTAPE consists of a series of pairs of records. The first record in the pair consists of a single word, written from common block /NWORDOUT/, which contains the number of words in the second record of the pair. The second record of the pair is written from array NHISTOUT of common block /HISTOUT/. The structure of the array NHISTOUT for the various possible events is shown in tables 1 through 7. The events which correspond to various event numbers (NHISTOUT(7)) are shown in table 8. Outcome codes for bombers and tankers (NHISTOUT(19),NBCODE) are shown in table 9. Outcome codes for missile launch (NHISTOUT(79),NTRYLC) are shown in table 10, and burst-damage codes (packed in NHISTOUT(20),INTARC) are shown in table 11.

After the NHISTOUT records have all been written out, 4HLAST is put on the file. The 400-word array IRECOV (from common block /RECOV/), which contains recovery base indices, is then written out. An end-of-file mark occurs next, followed by a record containing common block /BRKPNT/, a record containing common block /NAMES/, and then the YIELD array from common block /WARHEAD/.

Table 1. NHISTOUT Structure
(Bomber and Tanker Events)
(Sheet 1 of 2)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	NINBASE	Launch base index
3	NINDV	Vehicle index
4	NINTAR	Target index (INDEXNO)
5	NFUNC	Function
6	HTIME	Event time
7	NEVENT	Event type index
8	NPLACE	Event place (new zone)
9	NITYPE	Bomber type
10	NICLASS	Vehicle class
11	NIREG	Launch region
12	NIALERT	Initial alert status
13	NDUD	Warhead flag, 1 = OK, 2 = DUD*
14	NINZONE	Current (or old) zone
15	NNDECOYS	Number of decoys
16	NIALT	Current (or old) altitude
17	NNCM	Countermeasures index
18	NASMTYPE	ASM type**
19	NBCODE	Outcome code
20	NDEPEN	Recovery base***
21	THOUR	Time of abort****
22	FUTIME	Time of next event
23	--	Reserved for future use
24	--	Reserved for future use
25	NMHT	Number of lines in HT remaining
26	HDELAY	Delay (launch or other)

*LATTRIT only
**ALAUN only
***RECOVERY and RECHEK only
****BLAUN only

Table 1. (cont.)
(Sheet 2 of 2)

27 - 66	TINCN	TINCs of missed events
67 - 106	INDPN	INDPs of missed events
107 - 146	INDEN	INDEs of missed events
147 - 152	NWTYPEN	NWTYPE of unused warheads

Table 2. NHISTOUT Structure
(Missile Launch Event)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	--	*
3	--	*
4	--	*
5	NNWTYP	Warhead type
6	HTIME	Event time
7	NEVENT	Event type
8	NPLACE	Launch base index
9	NITYPE	Missile type
10	NICLASS	Vehicle class
11	NIREG	Launch region
12	NIALERT	Initial alert status
13	NNMIRV	Number of MIRVs
14	--	*
15	NNWPNS	Number of missiles
16	NNTARG	Number of targets
17	NNCYCLE	Number of missiles processed
18	NNSUCC	Number of missiles successfully launched
19	NNTEST	Number of missiles for reprogramming
20	--	*
21	--	*
22	TOTIME	Time on target for first missile
23	--	*
24	--	*
25 - 42	NWPNLIST	Missile indices
43 - 60	NLAUNLIS	Silo indices
61 - 78	NTARLIST	Target indices
79 - 96	NTRYLC	TRYLAUN outcomes

* Not used

Table 3. NHISTOUT Structure (Ballistic Missile Defense Events)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	NINBASE	Launch base index
3	NINDV	Vehicle index
4	NINTAR	Target index
5	NNWTYP	Warhead type
6	HTIME	Event time
7	NEVENT	Event type
8	NPLACE	Event place
9	NITYPE	Missile type
10	NICLASS	Vehicle class
11	NIREG	Launch region
12	NIALERT	Initial alert status
13	NNAL	Number of interceptors allocated (ballistic missile defense only)
14	Not used	
15	NNWHDS	Number of warheads (AREABMD only)
	NNDET	Number detonated (TERMBMD only)
16	NTAIM	Number of terminal aim points (AREABMD)
	NNPEN	Number of penetrators (TERMBMD only)
17	NNTERM	Number of warheads entering terminal defense (TERMBMD only)
18	NNAREA	Number of warheads entering area defenses (TERMBMD only)

Table 4. NHISTOUT Structure
(Complete Launch Event)

NHISTOUT	VARIABLE	DESCRIPTION
1 - 12		Same as BMD events
13	NMIRV	Number of MIRVs
14	NCODE	Outcome code (1 = success, 2 = failure)
15 - 28	NTLIST	MIRV target list

Table 5. NHISTOUT Structure
(Burst Damage Event)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	NINBASE	Launch base index
3	NINDV	Vehicle index
4	NINTAR	Target index
5	NNWTYP	Warhead type
6	HTIME	Event time
7	NEVENT	Event type
8	NPLACE	Event place
9	NITYPE	Vehicle type
10	NICLASS	Vehicle class
11	NIREG	Launch region
12	NIALERT	Initial alert status
13	NDGX	X-coordinate of desired ground zero
14	NDGY	Y-coordinate of desired ground zero
15	NDGZ	Desired height of burst
16	NAGX	X-coordinate of actual ground zero
17	NAGY	Y-coordinate of actual ground zero
18	NAGZ	Actual height of burst
19	NNCOL,ICUR	Number of collocated targets in group or current target in group
20 - 59	INTARC	Indices of collocated targets (right half of word)
20 - 59		Outcomes of collocated targets (left half of word)

Table 6. NHISTOUT Structure
(Time-dependent DBL Events)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	NINBASE	Base index
3	--	Not used
4	--	Not used
5	--	Not used
6	HTIME	Time of event
7	NEVENT	Event index
8	DTIME	Time of DBL(NAVCAL)
9	NITYPE	Vehicle type
10	NICLASS	Vehicle class
11	NIREG	Launch region
12	NIALERT	Alert status

Table 7. NHISTOUT Structure
(Zone Status Events)

NHISTOUT	VARIABLE	DESCRIPTION
1	NSIDE	Side
2	--	Not used
3	IBEG	Beginning zone index
4	--	Not used
5	IEND	Ending zone index
6	HTIME	Time of event
7	NEVENT	Event index (=6)
8 to 8 + IEND-IBEG	NPENZ	Number of penetrators in zone

Table 8. Event Numbers

EVENT	DESCRIPTION
1	Missile launch
2	Bomber or tanker launch
3	Missile complete launch
4	Bomber refuel
5	Enter zone
6	Zone status
7	Area attrition
8	Local attrition
9	Terminal ballistic missile defense
10	Burst damage
11	Tanker enter refuel area
12	Tanker leave refuel area
13	Bomber or tanker abort
14	ASM launch
15	Decoy launch
16	Recovery
17	Change altitude
18	Area ballistic missile defense
19	Check after recovery
20	Determine time of naval attrition
21	Naval attrition
22	Used for ENDGAME
23	Used for MONPRIN

Table 9. Bomber and Tanker Outcome Codes

NBCODE	DESCRIPTION
1	Success
2	Launch base dead
3	Takeoff abort
4	Refuel abort
5	No tankers available
6	Penetrate enemy territory
7	Leave enemy territory
8	Killed by area attrition
9	Killed by local attrition
10	Killed by local attrition after warhead release
11	ASM killed by local attrition
12	Tanker abort in refuel area
13	Departure of full tanker
14	Random abort
15	ASM launch abort
16	Not used
17	Recovery base dead on arrival
18	Scheduled splash
19	Abort on first of two refuelings
20	No tankers on first of two refuelings
21	No live recovery base on depenetration
22	Recovery base saturated
23	Killed after recovery
24	Killed after recovery at home base
25	Recovery at home base
26	Home recovery base dead on arrival
27	Arrival of ASM at target
28	Successful first of two refuelings
29	Refuel abort, alternate mission return home
30	No tankers available, alternate mission return home

Table 10. Missile Launch Outcome Codes

NTRYLC	DESCRIPTION
0	Missile not used
1	Not in commission
2	Silo dead
3	Launch abort
4	Silo destroyed on launch abort
5	Failure in powered flight
6	Successful launch
7	Missile planned for later launch event
8	Not in commission and silo dead

Table 11. Burst Damage Outcome Codes

CODE	DESCRIPTION
1	Target survives
2	No assessment necessary
3	Target killed
4	Target already dead
5	Target not in COLAR (error condition)

CONCEPT OF OPERATION

General

Each of the records read in from the EVENTAPE consists of a planned mission for a specific vehicle (bomber, tanker, or missile). Each mission consists of a series of events for that vehicle, each event to occur at a specific time. There are two major functions within SIMULATE:

1. To maintain the missions so that each event for each vehicle occurs at the proper time. This is done by maintaining the missions in an "event store" described below, in time order by the next event for each vehicle.

2. To simulate the occurrence of each event, with an appropriate subroutine.

After each event for a vehicle has been processed, if the vehicle still had additional events in its mission, the mission is reinserted into the event store so that its next event will be processed at the appropriate time. Then the event (mission) at the top of the list is processed. Thus the proper sequence of events is maintained. If at any time a vehicle is killed, the remaining mission is no longer returned to the list of future events.

Event Data

The data associated with a particular vehicle, when the next event for that vehicle is being processed, are stored in the array INDATA. The structure of this array varies with the type of event being simulated,* but in general consists of a series of simple variables and arrays describing the vehicle and its mission. Prior to putting the mission back into the event store, the event data (actually the data which will be placed into INDATA when the next event in the mission occurs) are placed into the array OUTDATA. Because many of the arrays within a particular OUTDATA structure are not used to capacity, not all of OUTDATA need be transferred to the event store. Therefore the data are first compressed according to the structure specifications for the event and placed into the array OUTDATAP. This is necessary to keep the size of the event store as small as possible. The data in OUTDATAP are placed, along with the time of the next event in the mission, the event type, and data structure specifications, into the event store in time sequence.

---

*See the description of common block /EDATA/ for the structure of INDATA for different type events.

When the next event for the vehicle is to be processed, the data for the event are taken from the event store and placed in the array INDATAP, which is in the same compressed form as when the same data were in OUTDATAP. The data are then placed into their proper locations in INDATA where they are available for processing, and the appropriate event subroutine is called.

Event Store

The event store, which contains all the future events to be processed, consists of two parts. The first part is kept in internal core memory in the form of a "list memory." When the available internal space is used up, the internal portion is put out (spilled) onto an external file. When more such spills have occurred than there are external files, the spilled events are merged with those on one of the files. The external portion of the event store thus consists of a number of files, each with a time-ordered list of events. At any given time, then, the next event to be processed may come from the internal event store or from any one of the external files.

List Memory

The sets of data associated with the missions of individual vehicles are kept in the event store in time order; thus at the head of the list is kept that mission data corresponding to the next occurrence to be simulated. When that occurrence or event has been processed, the remainder of the mission is inserted back into the list so that the next event for that vehicle will occur at the proper time. When such an insertion is made, it would be inconvenient if all missions whose next event is at a later time had to be shifted in the computer memory. To avoid this needless data processing, a list memory is used.

The list memory used by the Simulator consists of "cells" or groups of five words each. The scheduled mission for a vehicle consists of one "header" cell and a variable number of "data" cells. The five words of the header cell contain the following data, respectively:

1. The time of the next event in the mission
2. The type of event
3. Data packing specification
4. Link to the header cell of the next mission
5. Link to the first data cell

A data cell contains the link to the next data cell (or zero if there are no more data) plus four words of data. Thus when a new mission is to be inserted into the list, it is merely necessary to put the data into some available portion of the list memory, the physical location in the computer memory not being important; the various links are then adjusted to put the mission into its proper logical position.

SUMMARY OF SUBROUTINES PERFORMING MAJOR FUNCTIONS

Event Handling Subroutines

The calling hierarchy of the event handling subroutines is shown in figure 1. The flow of the event data is shown in figure 2.

Subroutine DONEXT: DONEXT determines the next event to be executed, either from internal list memory or from an external spill file, and calls the appropriate event subroutine. Entry EVSPILL empties the list memory onto an external spill file when list memory is filled.

Subroutine SQUEEZE: SQUEEZE fills array INDATAP with the event data in list memory.

Subroutine EVUNPK: EVUNPK fills array INDATA with the event data in array INDATAP, for use by the appropriate event subroutine. The transfer is made according to specifications associated with the event subroutine.

Subroutine PLANT: PLANT finds the appropriate place in list memory to put a new event (mission).

Subroutine EVPACK: EVPACK places the event data in array OUTDATAP.

Subroutine UNSQUEEZ: UNSQUEEZ transfers the event data from array OUTDATAP to internal list memory.

Event Simulation Subroutines

Subroutine ALAUN: ALAUN simulates the launch of an air-to-surface missile (ASM) by a bomber.

Subroutine AREABMD: AREABMD simulates the activity of an area ballistic missile defense against an incoming warhead and associated decoys.

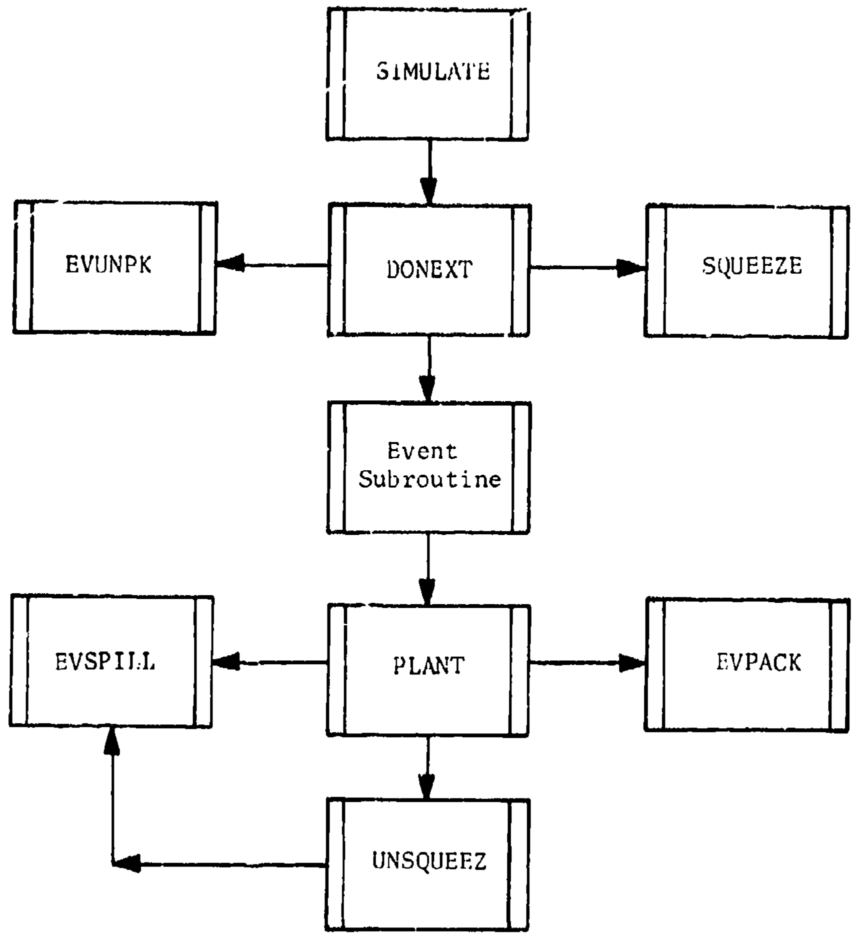


Fig. 1. Calling Hierarchy of Major Event Store Subroutines

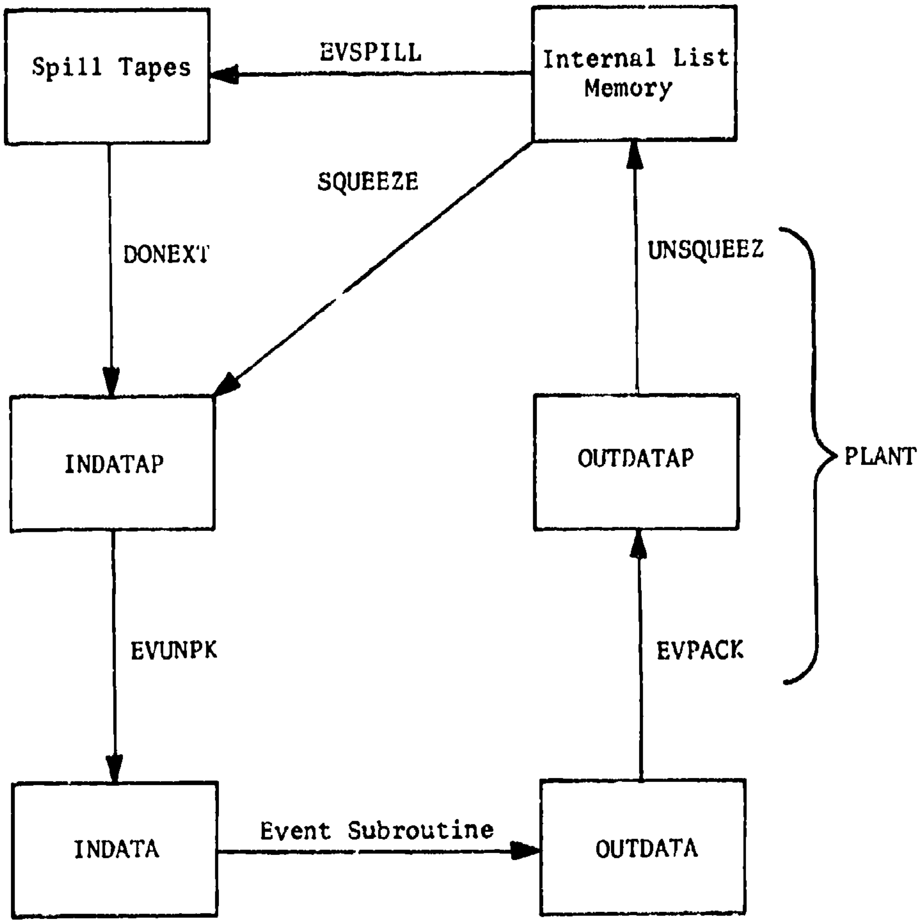


Fig. 2. Simulation Data Flow

Subroutine BABORT: BABORT simulates the scheduled or random abort of a bomber or tanker.

Subroutine BDAMAGE: BDAMAGE determines which targets are destroyed when a missile or bomber warhead detonates.

Subroutine BLAUN: BLAUN simulates the attempted launch of a bomber or tanker from its base.

Subroutine CHANGALT: CHANGALT simulates a change in bomber altitude between the two states high and low.

Subroutine CLAUN: CLAUN simulates the separation of MIRVs to their separate targets, and checks to see if the missile was destroyed during the powered flight phase.

Subroutine DLAUN: DLAUN simulates the launch or termination of bomber decoys.

Subroutine ERAREA: ERAREA simulates the arrival of a tanker at a refuel area.

Subroutine ESEC: ESEC simulates the entry and exit of a bomber from air defense zones.

Subroutine LATTRIT: LATTRIT simulates the entry of a bomber into the local defenses at a target.

Subroutine LRAREA: LRAREA simulates the departure of a tanker from a refuel area.

Subroutine MLAUN: MLAUN simulates the attempted launch of a group of missiles from a single squadron.

Subroutine NEXTEVNT: NEXTEVNT determines whether the next event for a bomber will be its regularly scheduled event, a random abort, or attrition from interceptor defenses.

Subroutine NAVATR: NAVATR simulates the kill of a target by naval attrition.

Subroutine NAVCAL: NAVCAL determines the time at which naval attrition will occur for a target.

Subroutine RECHEK: RECHEK tests to see if a bomber or tanker was killed by an enemy burst after recovery.

Subroutine RECOVERY: RECOVERY simulates the recovery of a bomber or tanker at its home or other base.

Subroutine REFUEL: REFUEL simulates the refueling of a bomber.

Subroutine TERMBMD: TERMBMD simulates the activity of a terminal ballistic missile defense against an incoming warhead or associated decoys.

Major Support Subroutines

Subroutine SIMULATE: SIMULATE initializes game conditions and reads in one or more event tapes.

Subroutine READIN: READIN reads in basic game data from the SIMTAPE prepared by the Data Input subsystem.

Subroutine RDCARDS: RDCARDS reads the data cards specifying various conditions for a particular simulation.

Major Output Subroutines

Subroutine AGZSUM: AGZSUM prints out at the end of a game a summary of the numbers and yields of delivered warheads by class and type of delivery vehicle.

Subroutine HIST: HIST prints out and records on tape the outcome from event subroutines.

Subroutine STATSUM: STATSUM prints out at the end of a game the original and final number of game items by class and type.

Subroutine SSTAT: SSTAT prints out at 15-minute intervals of game time the status of air defense zones.

COMMON BLOCK DEFINITIONS

External Common Blocks

The common blocks used by program SIMULATE in processing input/output (I/O) files are shown in table 12. Table 13 shows the structure of INDATA, and OUTDATA for the various events. Table 14 shows the structure of data in the STATUS array. Table 15 shows the structure of data in the COLAR array.

Internal Common Blocks

In addition to the common blocks associated with I/O operations, the common blocks described in table 16 are used internally by program SIMULATE. Table 17 indicates the common blocks used by the program subroutines described in subsequent sections of this chapter.

Table 12. Program SIMULATE External Common Blocks (Sheet 1 of 4)

INPUT FROM EVENTAPE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
EDATA	OUTDATA(320)	Structure of INDATA and OUTDATA depends upon event subroutine which uses it; INDATA blocks are read from EVENTAPE for launch events and naval DBL; see table 13 for an explanation of the structure of INDATA and OUTDATA as used for various events
	INDATA(320)	
RECOV	IRECOV(50,4,2)	Recovery base index
	TOFREC(50,4,2)	Time of flight to recovery
	NMORE(50,4,2)	Remaining recovery capacity
	NRES(50,4,2)	Remaining reservations
	LSHIFT	Unpacking constant

INPUT FROM SIMTAPE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
ASMS	PLABORT(20)	Probability of ASM launch abort
	CEPA(20)	Delivery error of ASM
BOMBER	PLABT(80)	Probability of bomber launch abort
	TMDEL(80)	Mean delay after abort
	ABRATE(80)	In-flight abort rate
	PRABT(80)	Probability of refuel abort
	CEPB(80)	Delivery error of bomber

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 12. (cont.)
(Sheet 2 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
BOMBER (cont.)	TLINTB(80)	Bomber launch spacing
	IFUNCB(80)	Bomber function
BRKPNT	NTYPECUM(15)	Cumulative number of types through end of a class
	NBLUETYP(15)	Number of Blue types in class
	INDBEGCL(15)	Beginning index of class
	INDBEGTY(250)	Beginning index of type
CAPACITY	DEFPOT(40)	Interceptor base type defensive potential
	CCPOT(40)	Command/control site type defensive potential
DAMAGE	WTR21MF	Maximum effects radius, squared
	VULN(63)	Target vulnerabilities
MISSILE	PINC(80)	In-commission probability
	PABORT(80)	Launch abort probability
	PDES(80)	Probability of destruction on abort
	PFPF(80)	Probability of failure in powered flight
	TPOWER(80)	Duration of powered flight
	TRET(80)	Time to retarget
	IR(80)	Reprogramming index
	CEPM(80)	Delivery error
	PKMIS(80)	Probability of destruction by ABM
	TLINTM(80)	Launch spacing
	IFUNCM(80)	Missile function

Table 12. (cont.)
(Sheet 3 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
NAMES	NAMESIDE(2)	Name of side
	NAMCLS(15)	Name of class
	NAMETYPE(250)	Name of type
NAVDATA	PK(10,10)	Probability of naval DBL
	TK(10,10)	Time of naval DBL
NCOL	NCOL	Number of collocated targets
	MAXIND	Total number of targets
PAYLOAD	MIRV(40,2)	Number of MIRVs
	IWTYP(40,2)	Warhead type
	NBOMB(40,2)	Number of MRVs
	NTDEC(40,2)	Number of terminal decoys
	NADEC(40,2)	Number of area decoys
TANKER	TPLABT(40)	Probability of tanker launch abort
	TTMDEL(40)	Mean delay after abort
	TABRATE(40)	In-flight abort rate
	TLINTT(40)	Launch spacing
	IFUNCT(40)	Tanker function
TBMDATA	PTK(2)	Probability of kill by terminal ABM
	NTINT(500)	Number of terminal interceptors
WARHEAD	YIELD(50)	Warhead yield
	PDUD(50)	Dud probability
	CEPW(50)	Spacing, if MRVs

Table 12. (cont.)
(Sheet 4 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
ZONES	AREA(63)	Area of zone
	ZDEFPOT(63)	Total interceptor potential in zone
	ZCCPOT(63)	Total command and control potential in zone
	NPENZ(63)	Number of penetrators
	KILLZ(63)	Number of kills by interceptor
	MAXZONE(2)	Maximum zone index for side
19501	NBCELL	Dimension of LINKM
	LINKM(7000)	Basic list memory
	STATUS(12000)	Target status (see structure, table 14)
	COLAR(4000)	Collocation data (see structure, table 15)

OUTPUT TO HISTAPE

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
HISTOUT	NHISTOUT(200)	See Output Files description for structure of NHISTOUT array for different events

Table 13. INDATA/OUTDATA Structure
(Sheet 1 of 5)

FOR BOMBER EVENTS

INDATA	VARIABLE	DESCRIPTION
1	SIDE	Side (Blue=1, Red=2)
2	INBASE	Launch base index
3	INDV	Vehicle index
4-8	-	Reserved for BDAMAGE
9	ITYPE	Bomber type
10	ICLASS	Vehicle class (bomber=2, tanker=3)
11	IREG	Launch region
12	IALERT	Alert status (alert=1, nonalert=2)
13	INZONE	Current defense zone
14	NDHI	Number of decoys at high altitude
15	NDLO	Number of decoys at low altitude
16	IALT	Altitude index (low=0, high=1)
17	NCM	Number of countermeasures
18	PKNAV	Target kill probability (if naval)*
19	-	**
20	-	**
21	MHT	Number of lines in History Table
22	MWT	Number of lines in Weapon Table
23	-	**
24	-	**
25	NPLAN	Number of lines in primary mission
26	TABORT	In-flight abort time

*PKNAV is INDATA(6) on EVENTAPE. Program SIMULATE transfers to INDATA(18).
**Reserved for future use.

Table 13. (cont.)
(Sheet 2 of 5)

INDATA	VARIABLE	DESCRIPTION
		History Table
27-106	TINC	Time increment
107-186	INDP	Event place
187-266	INDE	Event type
		Weapon Table
267-278	IWTYP	Warhead type
279-290	XDHZ	X-coordinate of desired ground zero
291-302	YDGZ	Y-coordinate of desired ground zero
303-314	ZDCZ	Desired height of burst

FOR MISSILE LAUNCH EVENT

1	SIDE	Side (Blue=1, Red=2)
2	INBASE	Launch base index
3	-	*
4	(TMLAUN)	**
5	IPAYLOAD	Payload index
6	-	*
7	-	*
8	NMIRV	Number of MIRVs/missile
9	ITYPE	Missile type
10	ICLASS	Vehicle class (missile=1)

*Not used
**Used only by SIMULATE

Table 13. (cont.)
(Sheet 3 of 5)

INDATA	VARIABLE	DESCRIPTION
11	IREG	Launch region
12	IALERT	Alert status (alert=1, nonalert=2)
13	NWPNS	Number of missiles
14	NTARG	2x [Number of targets]*
		Missile list
15-32	WPNLIST(I)	Missile indices
33-50	LAUNLIST(I)	Silo indices
51-320	MLIST(J)	Target data, Flight times **

FOR MISSILE COMPLETE LAUNCH EVENT

1	SIDE	Side (Blue=1, Red=2)
2	INBASE	Launch base index
3	INDV	Vehicle index
4	-	***
5	IPAYLOAD	Payload type
6	-	***
7	INITVULN	Silo vulnerability

* Maximum number of targets = 135. After EVUNPK, NTARG becomes number of target groups = total number of targets/NMIRV.

** Words appear in pairs, one for each target. The structure of the target data is:

15	12	12	12	3	3
INTAR	DGX	DGY	DHOB	Area Decoys	Terminal Decoys

*** Not used.

Table 13. (cont.)
(Sheet 4 of 5)

INDATA	VARIABLE	DESCRIPTION
8	NMIRV	Number of MIRVs
9	ITYPE	Vehicle type
10	ICLASS	Vehicle class
11	IREG	Launch region
12	IALERT	Alert status (alert=1, nonalert=2)
13-26	MDATA*	Target and coordinates
27-40	-	Times of flight

FOR BURST/DAMAGE,
AREA BALLISTIC MISSILE DEFENSE, AND
TERMINAL BALLISTIC MISSILE DEFENSE EVENTS

1	SIDE	Side (Blue=1, Red=2)
2	INBASE	Launch base index
3	INDV	Vehicle index
4	INTAR	Target index
5	NWTYP	Warhead type
6	DGX	X-coordinate of desired ground zero
7	DGY	Y-coordinate of desired ground zero
8	DHOB	Desired height of burst
9	ITYPE	Vehicle type
10	ICLASS	Vehicle class
11	IREG	Launch region
12	IALERT	Alert status (alert=1, nonalert=2)

*See structure of target data under Missile Launch event.

Table 13. (cont.)
(Sheet 5 of 5)

INDATA	VARIABLE	DESCRIPTION
13	NWHDS	Number of warheads*
14	TAIM	Number of terminal aim points*
15	AAIM	Number of area aim points (AREABMD only)**

NAVCAL EVENT

1	SIDE	1=Blue, 2=Red
2	INBASE	Base index
3	IDBL	Naval DBL index
4-8	Not used	-
9	ITYPE	Target type
10	ICLASS	Target class
11	IREG	Region
12	IALERT	Alert status

NAVATR EVENT

1	SIDE	1=Blue, 2=Red
2	INBASE	Base index
3	NCOL	Number of collocated targets
4-8	INCOL	Collocated target indices

*Not BDAMAGE
**Area BMD only

Table 14. Structure of Word in STATUS Array

VARIABLE	DESCRIPTION	BITS FROM RIGHT
TSTAT	1=alive, 0=dead	1
IKEEP	1 if alive/dead status is kept	3
TCOL	1 if collocated	4
NAVDBL	1 if NAVATR planted	6
TARDEFLO	Low-altitude defense potential	7-9
TARDEFHI	High-altitude defense potential	10-12
IATTACK	1 if selected for preferential defense	13
TVULN	Vulnerability index	16-21
IAREA	1 if in defended area	22
ITERM	Index to terminal BMD	25-33
KDEFZON	Area BMD zone	34-39
KDEFCMP	Area BMD component	40-42
ZONE	Bomber defense zone	43-48

Table 15. Structure of Word in COLAR Array

VARIABLE	DESCRIPTION	BITS FROM RIGHT
KEYDY	Y-distance to next target	1-11
KEYDX	X-distance to next target	12-22
KEYNTA	Number of targets in island	23-34
KEYTIN	Target index	35-48

Table 16. Program SIMULATE Internal Common Blocks
(Sheet 1 of 8)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
AATTRIT	FK	Scaling factor
	FCM	Effectiveness of countermeasures
	ALT	Effect of bomber altitude
	CON	Effect of command/control
	RPEN	Effect of number of penetrators
ABMDATA	IARDEF(2)	Preferential vs. random defense flag (1=preferential, 2=random)
	PSEL(2)	Probability of selection for defense (preferential); fraction of incoming objects to which interceptors are allocated (random)
	PAK(2)	Probability of killing a warhead
	PREM(2)	Probability of killed warhead being removed
	PAKD(2)	Probability of killing an area decoy
AGZSUM	NAGZ(2,80)	Number of warheads delivered
	TYD(2,80)	Total yield delivered
AREADAT	AINT(20,3)	Number of area interceptors
	NLRR(20)	Number of long-range radars
	IOVERLAP(20)	Packed radar data
	KRAD	Radar index

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 16. (cont.)
(Sheet 2 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CONST	LARGE	Large fixed point constant
	FTMAX	Large floating point constant
DATTA	ISIZE	Maximum INDATA element used
DEFZONE	DEFZONE	Area BMD zone
EPACK	INDFORMO(100)	Unpacked JFORMAT indices
EPSN	EPSN	Small constant
ERROR	IERR	Error type
ESTOR	SCALET	Not used
	SCALETI	Not used
	TMAX	Maximum game time
	NWORDS	Number of OUTDATAP words to output for planting event
	NDIMD	Dimension of INDATP and IOUTD
	NEXTEV	Pointer to next event in list memory
	MERGSW	Switch to indicate merging
	INDATP(320)	Packed INDATA
	IOUTD(320)	Packed OUTDATA
	ISVDAT(320)	Temporary packed data storage
	MEMTIM	Time of earliest list memory event--must immediately precede TAPET
	TAPET(10)	Earliest event time of each spill tape

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
ESTOR (cont.)	SVTIME(10)	Temporary time storage while merging
	ITOGO(10)	Words to go on each spill tape
EVENT	INDEV	Event index (see table 8)
	INDF	Event format index
EVINDX	NBIND	Number of sublists in list memory
	EVTIME(10)	Time of first event in sublist
	INDEVB(10)	Location of first event in sublist
FORMAT	JFORMAT(10)	Packing specifications

JFORMAT	Unused	U 9 Bits	D 12 Bits
Word Structure		INDATA Element Specifying Number Used	Dimension of Array

JFORMAT(I)

I	U	D	USED FOR
1	0	1	Simple Variables
2	0	0	Unused Variables
3	13	18	Missile Arrays (MLAUN)
4	14	270	Target Array (MLAUN)
5	22	12	Weapon Table Arrays
6	21	80	History Table Arrays
7	8	14	MIRV Arrays (CLAUN)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
FUTPRINT	IFUT	Option flag for printing FUTIME
GROUND	IAGX	X-coordinate of AGZ
	IAGY	Y-coordinate of AGZ
HISREC	IREC	Recovery base index
	IDEPEN	Depenetration corridor
	LAZONE	Depenetration zone
HISTABM	IAREAX	Index to area of current zone
	NAL	Number of interceptors allocated
	NWHDSX	Number of warheads penetrating area
	NDET	Number of objects detonating
	NAREA	Number of objects entering area defenses
	NTERM	Number of objects entering terminal defenses
HISTA1	NLO	Number of missiles left over
	TOFLT	Time of flight
	IW	Weapon index
HISTREF	NFGO	Units of fuel carried by departing tanker
	INEED	Units of fuel needed by bomber
INDEX	INDEX(30)	Randomly ordered indices
INBOMBF	INBOMBF	Option flag for using BOMBP

Table 16. (cont.)
(Sheet 5 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
INREP	INREP	Option flag for reprogramming
IPRINT	IPRINT(60)	Option flags for HIST prints
IPSWICH	-	INDATA print options
	IPSW1	Missile INDATA
	IPSW2	Bomber INDATA
	IPSW3	Tanker INDATA
	IPSW4	Missile targets, TOF
	IPSW5	Missile DGZ and decoys
	IPSW6	Bomber DGZ and warheads
	IPSW7	Naval attrition
KEYWORDS	KEYARRAY(20)	STATUS array unpacking keys
KEYWRDS	KTAR	Target unpacking key for array IOVERLAP
	KZON(3)	Zone unpacking keys for array IOVERLAP
LATTRIT	DEFHI	Defenders against high-altitude attack
	DEFLO	Defenders against low-altitude attack
	BKBRP	Probability of bomber kill before weapon release
LISTMEM	IAVAIL	Next unused cell in list memory
MONDAT	MONSW	Switch for diagnostic printout

Table 16. (cont.)
(Sheet 6 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
NEVTOT	NBEVTOT	Number of events in list memory
	NCTR	Number of events planted since last directory update
NSW	[illegible]W	Number of MONSW data cards
NWORDOUT	[illegible]RDOUT	Number of NHISTOUT words written out in current block
PLANTS	MLFIA(10)	Missile launch packing formats
	BLFIA(10)	Bomber event packing formats
	CLFIA(10)	Complete launch packing formats
	TBFIA(10)	Terminal BMD packing formats
	ABFIA(10)	Area BMD packing formats
	NAFIA(10)	Naval attrition packing formats
	SSFIA(10)	Zone status packing formats
READ	HHOUR	Game time H-hour
	NET	Number of event tapes used
	DELAY(2)	Delay for each side
REFUEL	NFTANK(100)	Number of units of fuel in area
	NETANK(100)	Capacity of empty tankers in area
TBMDATA	PTK(2)	Probability of kill by terminal BMD
	NTINT(500)	Number of terminal interceptors
TIME	TIME	Current game time
	FUTIME	Time of event being planted

Table 16. (cont.)
(Sheet 7 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TRYL	INDEXWPN(18)	Weapon indices of successful launches
	PINCOMM	In-commission probability
	PNOABORT	Probability of not aborting
	PDEST	Probability of destructive abort
	PFLTFAIL	Probability of powered flight failure
	TPF	Time required to leave lethal radius of silo
	TRETARG	Time required to retarget
	IREP	Reprogramming capability
	NCYCLE	Number of missiles processed
	NCOM	Number of missiles in commission
	NALIVE	Number of missiles alive
	NNABT	Number of missiles not aborted
	NSUCC	Number of missiles successfully launched
	NLATER	Number of missiles to be launched later
	NDES	Number of missiles destroyed
	NTEST	Number of missiles for reprogramming
VULNDATA	PG(12)	Data for P vulnerability, ground burst
	PA(12)	Data for P vulnerability, air burst
	QG(8)	Data for Q vulnerability, ground burst
	QA(8)	Data for Q vulnerability, air burst

Table 16. (cont.)
(Sheet 8 of 8)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
ZSTATUS	JTMAX	Maximum number of zone status printouts

Table 17. Common Blocks Used by SIMULATE Subroutines (Sheet 1 of 2)

COMMON BLOCKS

SUBROUTINES	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
AAFIRIT														x																				
ADDEM			x																															
AGZNUM							x																											
ALAIN					x									x																				
AILASMD		x		x									x	x												x				x				
EAROBT														x																				
LOAMING			x	x	x		x	x			x			x										x									x	
BLADE						x						x		x																				
PHISTAT		x		x			x																											
BOMBT														x														x			x			
BEESPRIN														x																				
CHANGALE														x																				
CLAUS		x												x																				
DEAUS														x																				
DONEAT										x								x	x	x														
ENDGAME							x																											
FRAIGA												x		x		x																		
ERROE																																		
ESEC														x		x									x			x						
EVPACK														x	x			x				x												
EVUNPK														x	x			x				x												
HIST				x		x	x						x	x										x	x	x	x	x	x				x	
HISTWRIT																												x						
IFIND																																		
INITLV										x								x																
INITLIST																		x		x														
LATTSIT					x	x								x										x										
LRABLA														x		x												x	x					
MLAUS														x													x			x		x		
MSPRIN														x					x															
MSTSPRIN														x																				
MUSEK														x																				
NAVATR														x																				
NAVCAL														x																				
NEXTEVNT	x											x		x		x																		
PLANT																		x	x	x														
PLANTS												x		x					x				x											
PRINPAT														x				x																
PROGDAT														x				x																
RANORDER																														x				
RECARDS	x										x												x								x	x	x	
READIN				x	x	x	x	x	x		x										x													
RECHEK														x											x									
RECOVERY														x											x			x						
RETURL						x								x														x						
[illegible]																		x																
RNQLA																																		
SIMULATE		x	x	x		x	x					x		x				x	x		x	x	x					x					x	
SQUEEZ																		x		x														
SSTAT																												x					x	
STATSUM							x																											
TERMSD														x										x		x				x				
TRYLAGN														x																				
UNCLOG															x				x															
UNSQUEEZ																		x																
UPDER																		x		x														
VERAD																																		
ZAPGET																																		

Table 17. (cont.)
(Sheet 2 of 2)

COMMON BLOCKS

SUBROUTINES																																		
AXTIRIT																														x				
AGROOM						x																												
AGASOM											x																							
ALMGN																										x								
ARLAROS			x																														x	
BABORT																										x					x			
BDAMAGE			x	x			x				x						x													x	x		x	
BLAUN			x																					x		x								
BMDSTAT			x								x														x								x	
BOBILE																	x																	
BUESPRIN	x																																	
CHANGALT												x																						
CLAUN			x				x											x								x							x	
DLAUN																										x					x			
DOSEXT								x	x					x												x		x					x	
ENDGAME		x							x		x										x					x		x		x				
ERAREA																						x				x								
ERROR																																		
ESEC			x																		x					x					x		x	
EVPACK																																		
EVUNPK								x																										
HIST			x				x	x			x						x									x	x			x				
HISTWRIT		x															x									x		x						
IFIND													x																				x	
INITEV																										x								
INITLIST						x		x																									x	
LAFIRIT			x		x																					x				x	x		x	
LRAREA			x																		x	x				x							x	
MLAUN			x				x																			x							x	
MONPRIN								x								x	x									x								
MSTSPRIN	x																																	
MUNPK																																		
NAVATR			x																														x	
NAVCAL			x									x					x									x							x	
NEXTEVNT																										x					x			
PLANT						x								x												x							x	
PLANTS																										x								
PRINDAT																										x								
PROUTDAT																										x								
RANORDER																																		
RECVRDS	x				x						x					x									x				x		x			
READIN							x					x	x					x		x			x	x	x	x		x		x	x			
RECHECK			x														x				x												x	
RECOVERY			x																							x							x	
RETRIEL																						x				x								
RETIM						x																											x	
RNTEV																																	x	
SIMDATA	x	x	x	x					x	x	x		x		x		x		x	x	x	x	x	x		x		x			x		x	
SQUELCH																																		
SSTAT											x						x									x					x	x		
STATSUM			x								x																						x	
TEAMEND			x				x																		x								x	
TEVENDS			x																								x							
UNPTOR																			x															
UNSQUEEZ																																	x	
UPDUE														x																			x	
VIRSO																													x					
ZVOCT		x							x												x							x						

PURPOSE:	To initialize arrays, read the event data, plant the initial events, and start the simulation.
ENTRY POINTS:	SIMULATE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AGZSUM, AREADAT, BOMBER, BRKPNT, DATTA, EDATA, ESTOR, EVENT, FILABEL, FORMAT, FUTPRINT, HISTOUT, IPRINT, IPSWICH, ITP, KEYWRDS, KEYWORDS, MYIDENT, MYLABEL, NAMES, NCOL, NOPRINT, NWORDOUT, PLANTS, READ, RECOV, REFUEL, STATUS, TANKER, TIME, TWORD, ZONES, 19501
SUBROUTINES CALLED:	ABORT, ADDMEM, BTINPRIN, BMDSTAT, DONEXT, INITAPE, INITEV, INITLIST, KEYMAKE, MSINPRIN, OPTPRIN, PAGESKP, PLANT, PLANTS, RANFSET,* RDARRAY, RDCARDS, RDWORD, READIN, SETERROR, SETREAD, SETWRITE, TERMTAPE
CALLED BY:	None

The operations performed by program SIMULATE are shown in figure 3. As indicated (statement 1), SIMULATE begins by initializing several arrays. All the words in arrays INDATA, OUTDATA, STATUS, COLAR, NPENZ, KILLZ, NAGZ, TYD, NFTANK, NETANK, NHISTOUT and IRECOR are set to zero. Then, subroutine INITAPE is called to initialize the file handling operations. The order in which the event spill files are used is determined by filling the first N words of ITAPES, where N is the number of spill files. ITAPES(N+1) is set to zero. Subroutine INITEV is called to initialize the event handling subroutines, and INITLIST to initialize list memory.

Next, the STATUS array is divided into the following packed items through calls on KEYMAKE: TSTAT, IKEEP, TCOL, NAVDBL, TARDEFLO, TARDEFHI, IATTACK, TVULN, IAREA, ITERM, KDEFZON, KEDFCMP, and ZONE. Each COLAR array word is divided into the following packed items through calls on KEYMAKE: KEYTIN, KEYNTA, KEYDX, and KEYDY. Each IOVERLAP array word is divided into the following packed items through calls on KEYMAKE: KTAR, KZON(1), KZON(2), and KZON(3).

*System Library Function

The Simulator input data cards are read and printed by a call to RDCARDS.

ITP, current tape unit, is set to seven to indicate the simulation data tape SIMTAPE and subroutine READIN is called to read the tape.

Subroutine BMDSTAT is called to print the ballistic missile defense status. IMAX is set to the index of the first word of array STATUS beyond the last word used by the final type of item. NMAX is set to the number of unused words, and ADDMEM is called to add this unused portion of STATUS to list memory. IMAX is set to the index of the first word of array COLAR beyond the last word used for collocated targets. NMAX is set to the number of unused words, and ADDMEM is called to add this unused portion of COLAR to list memory.

KPASS is set to one. All of the following, through the call on subroutine TERMTAPE, is done for each event tape. (The Simulator can perform a one or two sided simulation. An event tape is required for each plan being simulated.) ITP is set to six, the event tape. Subroutine SETREAD is called to put tape ITP in read status.

ITP is set to six and subroutine RDWORD called to read a data word. When it is zero, control goes to statement 32. If not, RDARRAY is called to read an INDATA block. The record is checked to determine whether a NAVDBL event is called for; if it is, it is planted. Otherwise ICLASS, vehicle class, is tested. If ICLASS is one, the vehicle class is "missile." FUTIME, event execution time, is set to the DELAY for the current side plus the planned launch time. Subroutine PLANTS is called to plant a missile launch event for time FUTIME. IPSW1 is tested. If it is one, the event data block is printed in expanded form by a call to MSINPRIN. In any case, control is then transferred to statement 20.

If ICLASS is two, the vehicle is a bomber. The spacing factor TLINT is set to the spacing factor for this bomber type TLINTB(ITYPE). If ICLASS is three, the vehicle is a tanker. The spacing factor TLINT is set to the spacing factor for this tanker type TLINTT(ITYPE). In either case, event execution time is set to the first History Table time increment TINC(1), plus the delay time for this side DELAY(SIDE), plus TLINT, plus the time to become invulnerable to a weapon burst at the base. The History and Weapon tables are inverted so that the first event is at the bottom of the list. PLANTS is called to plant a bomber/tanker launch event for time FUTIME. If desired, the event data block is printed in an expanded form by a call to BTINPRIN. In any case, control is transferred to statement 20.

When the last event record on the current tape has been read, the recovery arrays are read, and subroutine TERMTAPE is called to terminate the current event tape. When the required number of event tapes has been read, ITP is set to five to indicate the History tape, and subroutine SETWRITE is called to put the tape in write status.

The first zone status event is planted for time zero, and the end-of-game event is planted for time TMAX, maximum game time, minus .01 hours. PAGESKP is called to eject a page on the printer, and subroutine DONEXT is called to start and control the simulation.

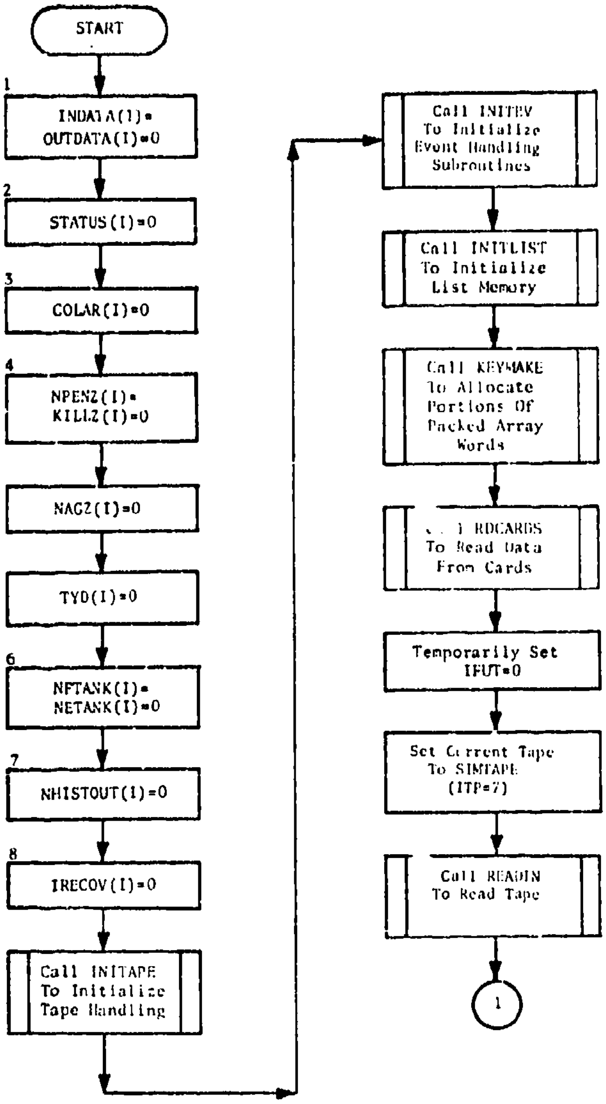


Fig. 3. Program SIMULATE
(Sheet 1 of 4)

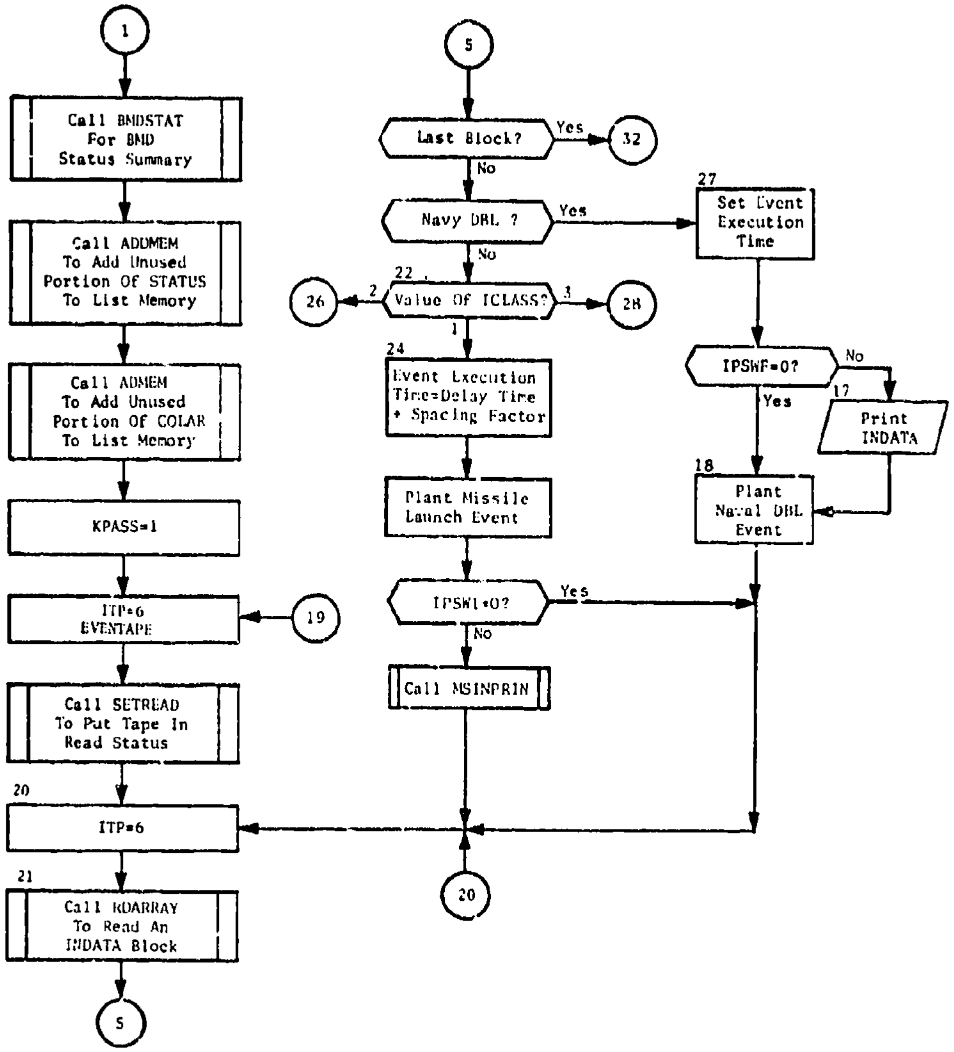


Fig. 3. (cont.)
(Sheet 2 of 1)

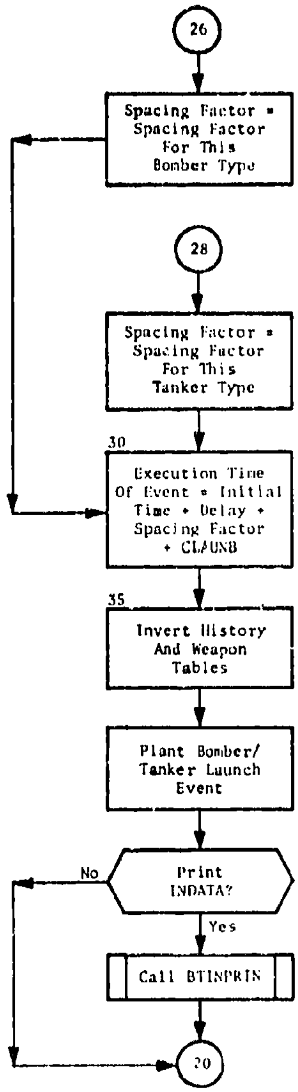


Fig. 3. (cont.)
(Sheet 3 of 4)

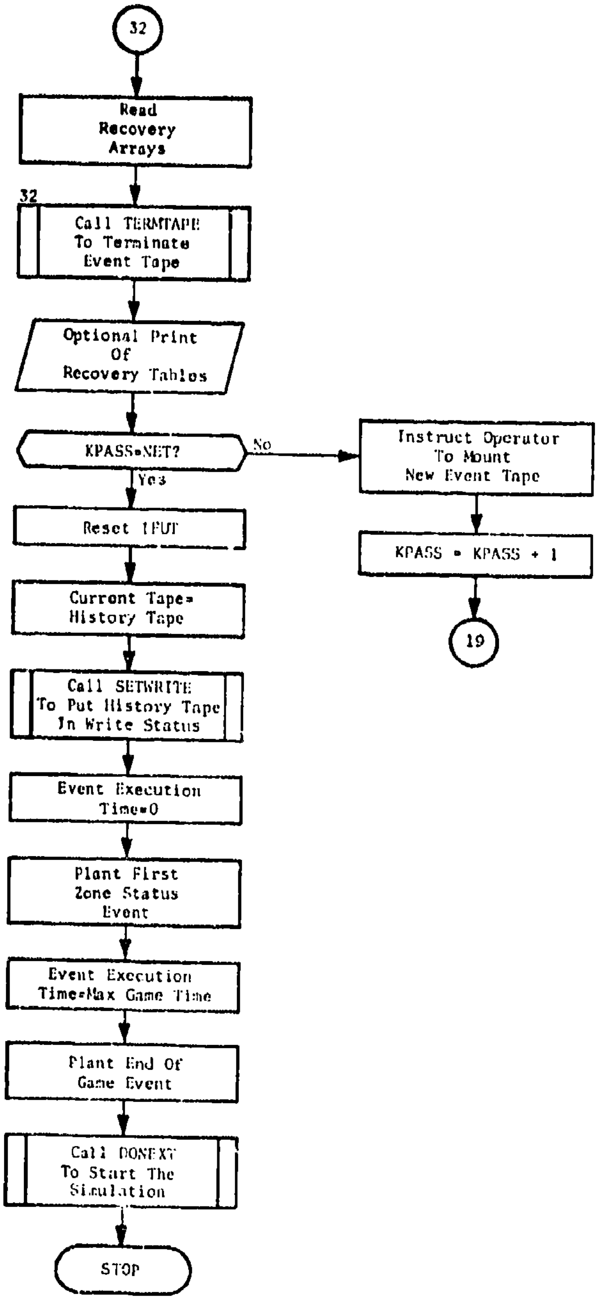


Fig. 3. (cont.)
(Sheet 4 of 4)

SUBROUTINE AATTRIT

PURPOSE:	1. To record bomber kill by enemy area defense.
	2. To decrease the number of penetrators in the defense zone.
ENTRY POINTS:	AATTRIT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, ZONES
SUBROUTINES CALLED:	HIST
CALLED BY:	DONEXT

Method

Relevant data are transferred to the array NHISTOUT by a call on subroutine HIST. Additional recording functions are performed by a second call on HIST. The number of penetrators in the zone, NPENZ(INZONE), is decreased by one for the bomber and by the number of decoys NDHI and NDLO at high and low altitude, respectively. The number of bombers killed by area attrition in the current zone, KILLZ(INZONE), is increased by one, and the subroutine exits.

Subroutine AATTRIT is illustrated in figure 4.

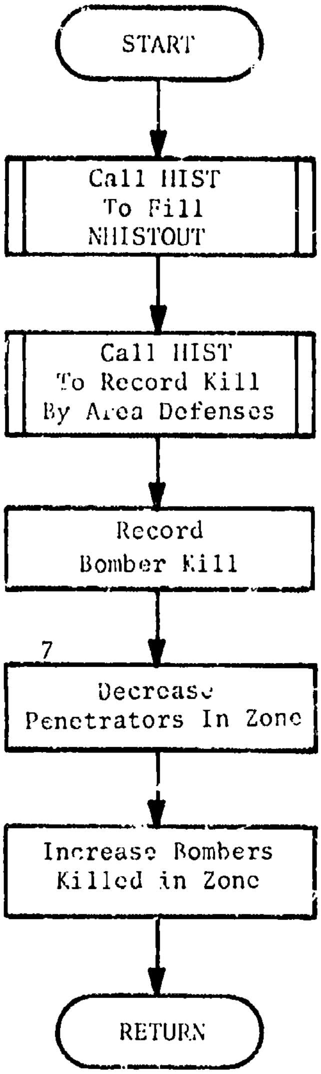


Fig. 4. Subroutine AATTRIT

SUBROUTINE ADDMEM

PURPOSE: To add an array to available list memory.

ENTRY POINTS: ADDMEM

FORMAL PARAMETERS:
- IND - Starting index in the array IARR
- NBWDS - Number of words to be added
- IARR - Array to be added

COMMON BLOCKS: LISTMEM

SUBROUTINES CALLED: None

CALLED BY: SIMULATE, INITLIST

Method

The counter NC is set to zero, and I is initialized to IND. If NBWDS is less than five, the number of words in a list memory cell, the subroutine returns.

Links are stored in every fifth word of IARR, and NC incremented by five until NC exceeds NBWDS. Then the last cell is linked into list memory, and the first cell is labelled as the next available cell of list memory. The array IARR must be in common block /LINL1/.

Subroutine ADDMEM is illustrated in figure 5.

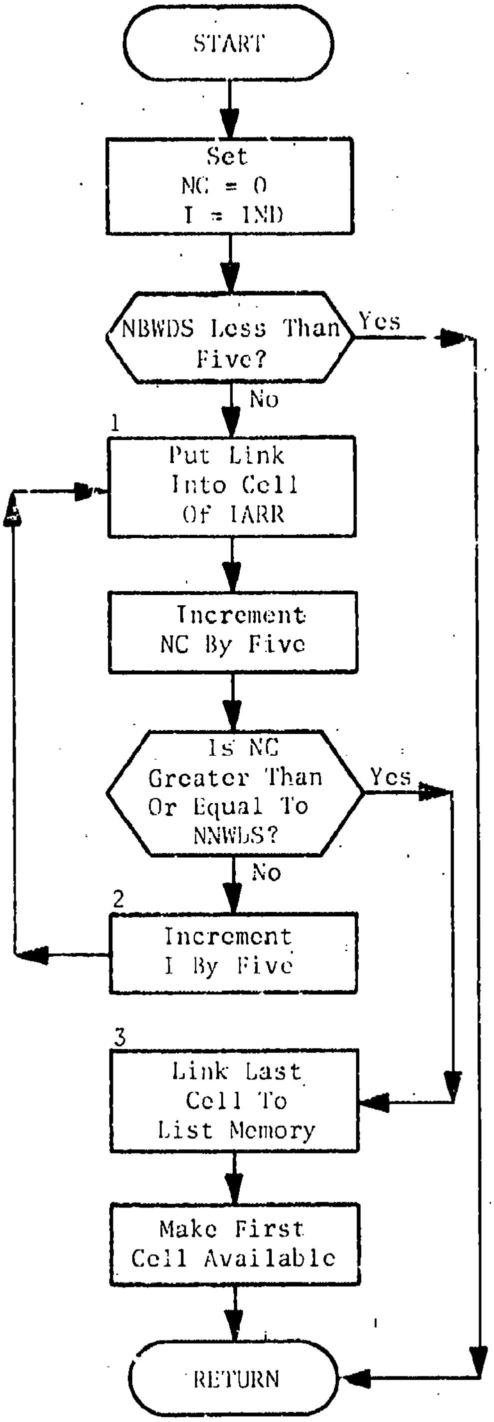


Fig. 5. Subroutine ADDMEM

SUBROUTINE AGZSUM

PURPOSE:	To summarize at the end of the simulation the number of warheads delivered and their total yield, broken down by side, class, and type of delivery vehicle.
ENTRY POINTS:	AGZSUM
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AGZSUM, BRKPNT, NAMES
SUBROUTINES CALLED:	None
CALLED BY:	ENDGAME

Method

The number of warheads delivered and their yield are accumulated in subroutine BDAMAGE and stored by side and type of delivery vehicle. Warheads delivered by ASM are included under the bomber type which launched the ASM.

First the Blue summary is printed out, then the Red. Within each side, missiles are printed first, then bombers.

In addition to the listings by type of delivery vehicle, there is a missile total, a bomber total, an overall total for each side, and a summary total in the game.

Subroutine AGZSUM is illustrated in figure 6.

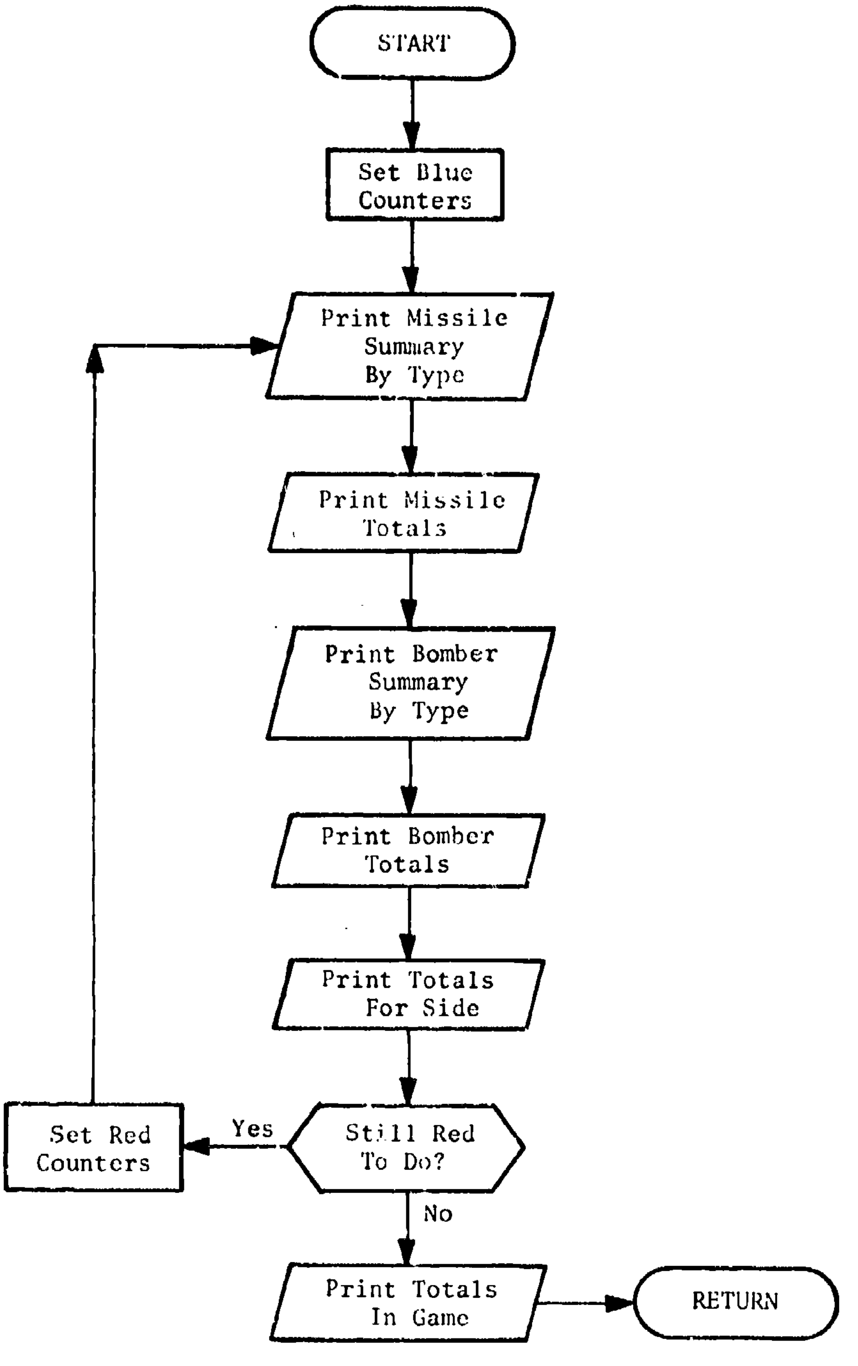


Fig. 6. Subroutine AGZSUM

SUBROUTINE ALAUN

PURPOSE:	To simulate the attempted launch of an air-to-surface missile (ASM) from a bomber.
ENTRY POINTS:	ALAUN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ASMS, EDATA, TIME
SUBROUTINES CALLED:	HIST, NEXTEVNT, PLANTS, RANF*
CALLED BY:	DONEXT

Method

The execution of ALAUN is recorded through a call to subroutine HIST. The ASM type JTYPE is found in the bomber History Table entry INDP(IHT), and the launch abort probability for that type PLABORT(JTYPE) is retrieved. This probability is compared to a random number generated by library subroutine RANF to determine if the ASM is successfully launched.

If the launch is unsuccessful, this is recorded.

If the launch is successful, the event data for the ASM are moved from INDATA to OUTDATA. The data moved include side, vehicle index, weapon type, event type, event time, event place, and desired ground zero components. ASM type is stored in the countermeasures cell. A Local Attrition event is then planted for the ASM through a call to subroutine PLANTS, and the successful launch is recorded.

In either case the History Table pointer IHT and the Weapon Table pointer JWT are moved, and the next bomber event is prepared for by calling subroutine NEXTEVNT.

Subroutine ALAUN is illustrated in figure 7.

---

*System Library Function

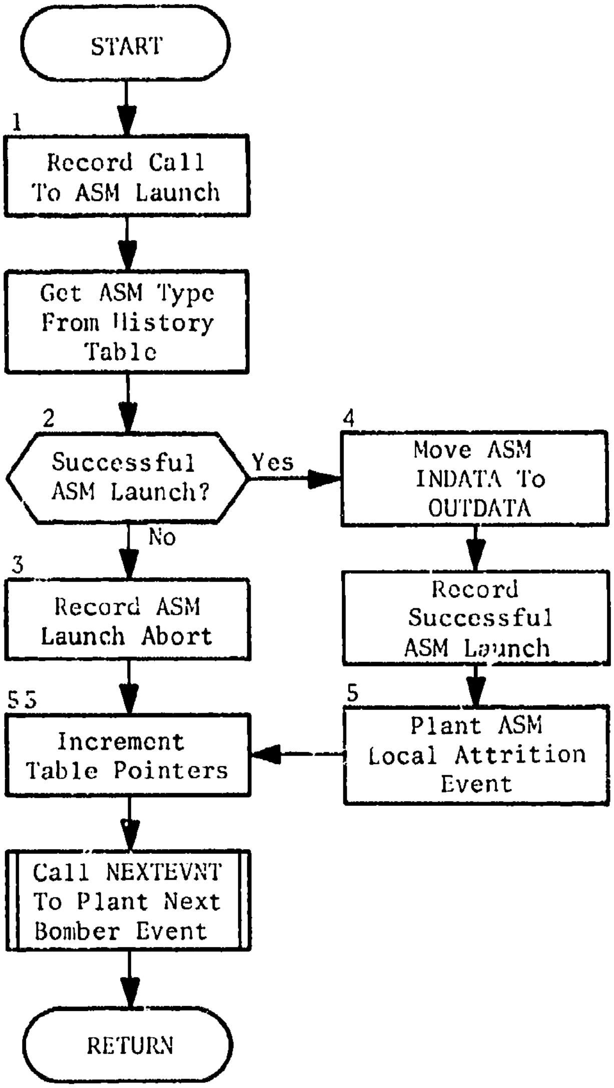


Fig. 7. Subroutine ALAUN

PURPOSE: To test for survival of missiles against an area ballistic missile defense system.

ENTRY POINTS: AREABMD

FORMAL PARAMETERS: None

COMMON BLOCKS: ABMDATA, AREADAT, DEFZONE, EDATA, HISTABM, INDEX, KEYWORDS, 19501

SUBROUTINES CALLED: HIST, HISTWRIT, IFIND, IGET, IPUT, RANF,* RANORDER, TERMBMD

CALLED BY: DONEXT

Method

Subroutine AREABMD (see figure 8) simulates the activity of an area ballistic missile defense against an incoming warhead and its associated decoys.

In QUICK, up to 20 ballistic missile defense areas may be established. Each of the defended areas may have up to three interceptor bases, each with its own inventory of antiballistic missiles. There may be as many as three long-range radars supplying surveillance information to the interceptor sites.

When called, subroutine AREABMD computes OSIDE, the defending side. This is accomplished by subtracting the value SIDE from three; i.e., OSIDE=3-SIDE. IAREA, the area defensive status of the current target, and KDEFZON, the area defense zone of the target, are retrieved through calls on IGET and stored in IA and DEFZONE, respectively. If IA is zero, if NLRR(DEFZONE) is zero, or if all the ABM bases in the zone are exhausted (AINT(DEFZONE,I)=0,I=1,2,3)), AREABMD is skipped and TERMBMD is called.

Otherwise NINTR(DEFZONE) is set equal to the total number of area interceptors in DEFZONE for each area defense zone, HIST is called to record the call to AREABMD, and counters NPEN, NDEC, NAL, and KCOL are set to zero. IAREAX is set to IA and NWHDSX to NWHDS. IARDEF(OSIDE), type of area defense for this side, is tested. If the value is one, the defense will be preferential. If the value is two, the defense will be random.

---

*System Library Function

Preferential Defense

If a preferential defense is used, the attack indicator IATTACK, indicating if the target has already been attacked, is retrieved through a call to IGET and stored in IAT. If IAT is nonzero, the target has been attacked. This is recorded and control transferred to statement 9 below. Otherwise PSEL(OSIDE) is compared to a random number to see if the target will be defended. If so, ISEL is set to one, and the selection is recorded. If not, ISEL is set to zero, and the non-selection is recorded. Control passes to statement 7 in either case.

In statement 7, IATTACK, the preferential attack indicator, is turned on and IAREA, the area defensive status, is set to the value of ISEL through calls on subroutine IPUT. KCOL, the collocation flag, is tested to see if the current target is the primary target of the weapon. If so, TCOL, collocation indicator for this target, is retrieved through a call to IGET. If KOLOC is zero, control is transferred to statement 8 below. Otherwise INTAR, the target index, is saved in INTO and KCOL is set to one.

Subroutine IFIND is called to find the index of the word in array COLAR containing the collocation information for INTAR and to store the index in KIND. KEYNTA, the number of targets in the collocated group, is retrieved through a call to IGET and stored in NTA. If NTA is nonzero, the next target is NTA targets back in COLAR, so KIND is decreased by NTA. In either case, KIND is increased by one. KEYTIN, the next target index, is retrieved through a call to IGET and stored in IX. If IX equals INTO, the list is complete and control is transferred to statement 8 below. Otherwise, control is transferred to statement 7 above.

In statement 8, if ISEL is zero, the target is undefended, and control is transferred to statement 21 below. If ISEL is one, control passes to statement 9 below.

In statement 9, AINT(Z,1), AINT(Z,2), and AINT(Z,3), the number of interceptors at each of the ABM bases covering area defense zone Z, are compared and a permutation of the integers 1, 2, 3 is stored in IORD1, IORD2, and IORD3 in such a way that AINT(Z,IORD1)$\geq$AINT(Z,IORD2)$\geq$AINT(Z, IORD3). NINTR(Z), the total interceptor supply at area defense zone Z, is compared to AAIM, the number of area aim points. If AAIM is smaller, control is transferred to statement 32, below. Otherwise, NINTR(Z) is reduced by AAIM; NAL, the number of defenders allocated, is set to AAIM; and AINT(Z,I), I = 1, 2, 3 are reduced so as to simulate firing interceptors, first from the ABM base with the largest interceptor supply, then from the base with the next largest supply, and so forth until NAL interceptors have been fired.

In statement 12, the following checking is done for each of the NWHDS warheads: PAK(OSIDE), probability of area kill, is compared to a random number. If PAK(OSIDE) is smaller, the number of penetrators NPEN is increased by one, and the next warhead is tested. Otherwise, PREM(OSIDE), the probability of warhead removal, is compared to a random number. If PREM(OSIDE) is smaller, the number of decoys NDEC is increased by one. In either case, the next warhead is tested.

When each of these warheads has been tested, ND is set to the difference between the number of terminal aim points TAIM and NWHDS. The following check is made for each of the ND decoys: if PAKD(OSIDE), the probability of area kill of decoy, is smaller than a random number, NDEC is increased by one to indicate terminal decoy leakage.

When each of these decoys has been tested, NWHDSX and NWHDS are each set to NPEN, and TAIM is set to the sum of NPEN and NDEC.

In statement 21, the number of attacking objects that have leaked through is recorded, subroutine HISTWRIT is called to write out the HISTOUT block for the AREABMD event, subroutine TERMBMD is called, and the subroutine exits.

Random Defense

In statement 28, the following test is made for each of the AAIM area aim points. If PSEL(OSIDE), the probability of selection for defense, is not less than a random number, then the number of defenders allocated NAL is increased by one. When each of these points has been tested, control is transferred to statement 32.

In statement 32, NINTR(Z) is compared to NAL. If NAL is smaller, NAL is set to NINTR(Z) and the allocation of interceptors is recorded. In either case, NINTR(Z) is reduced by NAL, and AINT(Z,1), AINT(Z,2) and AINT(Z,3) are reduced so as to simulate firing the weapons, first from the base with the largest weapon supply, then from the base with the next largest, and so forth until NAL weapons have been fired. The number of objects not attacked, NREM, is set to AAIM minus NAL.

ICAT is initialized for randomizing. Words one through NWHDS are set to one; words NWHDS+1 through TAIM are set to two; and words TAIM+1 through AAIM are set to zero. Subroutine REORDER is called to randomly reorder INDEX. Words INDEX(1) through INDEX(REM) of ICAT are tested. If the value is one, the object is a penetrator; if two, a terminal decoy. After these NREM words are tested, NWHDS is decreased by NPEN, TAIM is decreased by the sum of NDEC and NPEN, and control is transferred to statement 12, above.

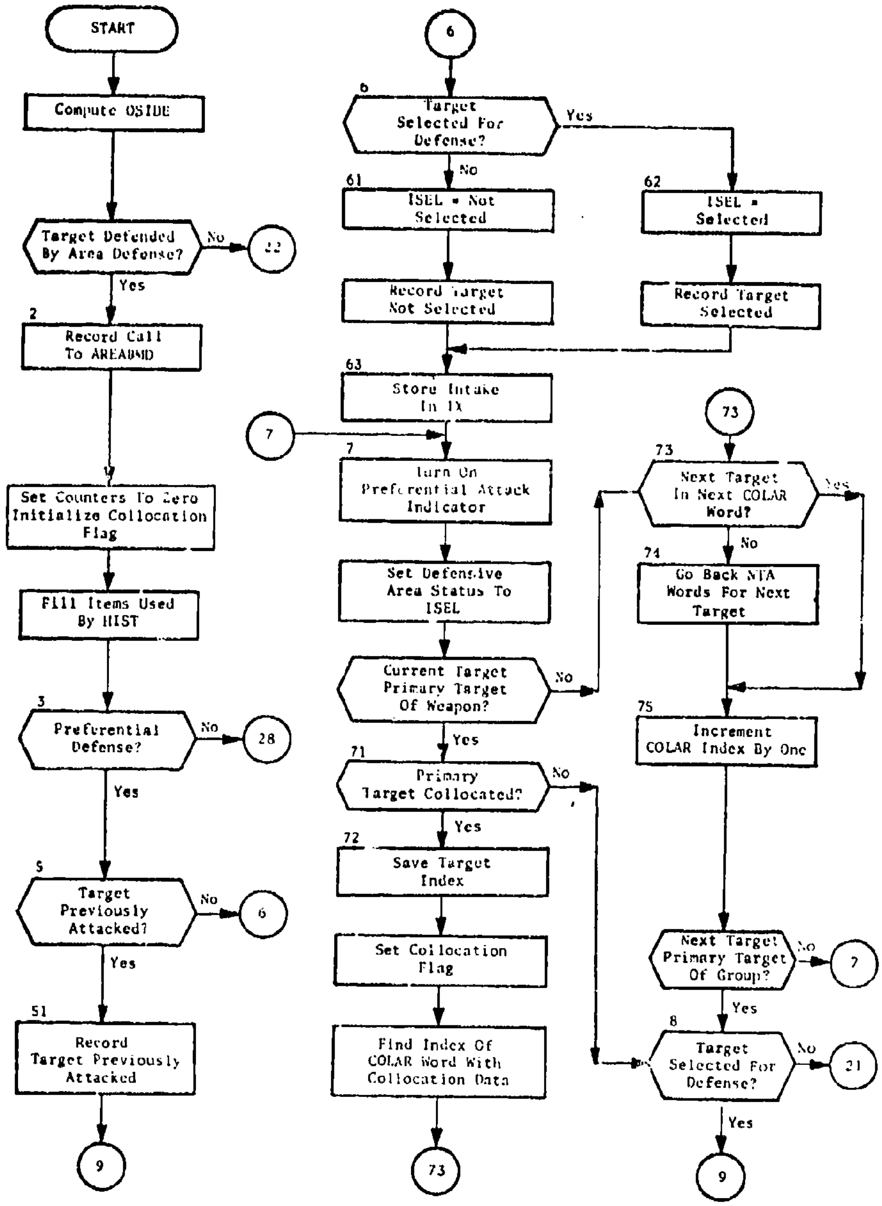


Fig. 8. Subroutine AREABMD
(Sheet 1 of 2)

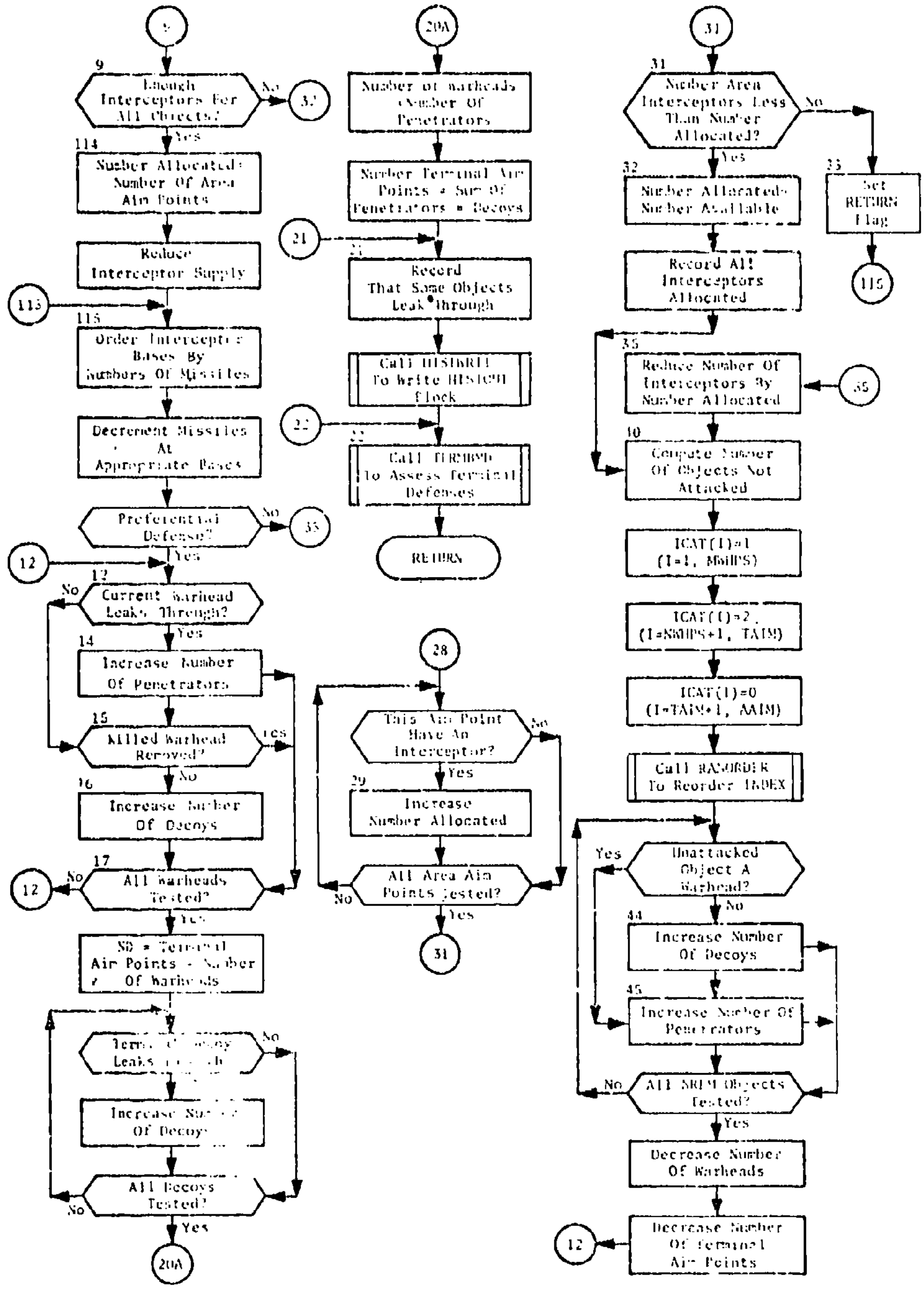


Fig. 8. (cont.)
(Sheet 2 of 2)

SUBROUTINE BABORT

PURPOSE:	To record a bomber abort and to decrease the number of penetrators if the bomber aborts after entering enemy territory.
ENTRY POINTS:	BABORT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, TIME, ZONES
SUBROUTINES CALLED:	HIST
CALLED BY:	DONEXT

Method

If the current zone INZONE is in enemy territory and the vehicle is not a tanker, the number of penetrators in that zone NPENZ(INZONE) is decreased by one (for the bomber) plus the number of decoys the bomber has launched.

If the current event index in the bomber History Table is equal to 13 (bomber abort), the abort is recorded as a scheduled splash. Otherwise, it is recorded as a random abort.

Subroutine BABORT is illustrated in figure 9.

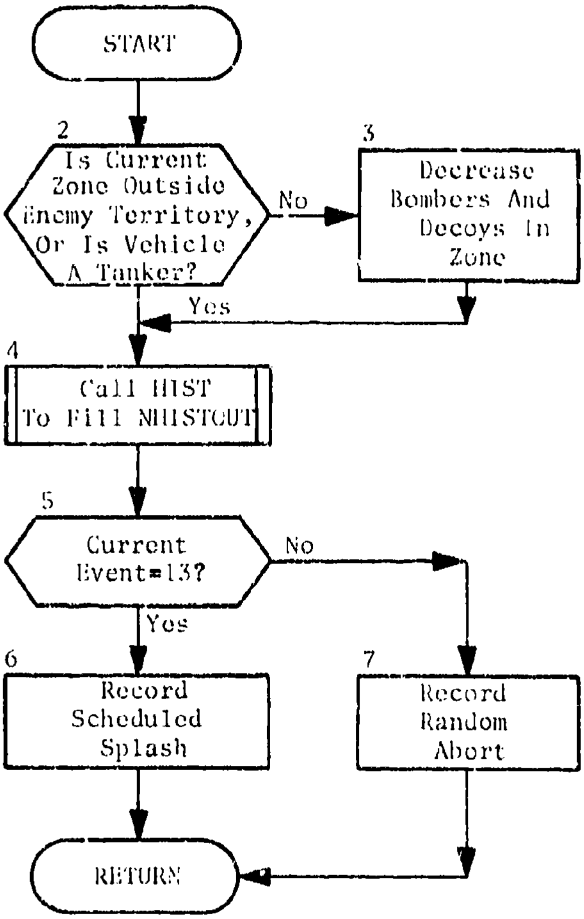


Fig. 9. Subroutine BABORT

SUBROUTINE BDAMAGE

PURPOSE:	To calculate burst damage to a target or a collocation group of targets.
ENTRY POINTS:	BDAMAGE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AGZSUM, AREADAT, ASMS, BOMBER, BRKPNT, CAPACITY, DAMAGE, EDATA, GROUND, IPRINT, KEYWORDS, KEYWRDS, MISSLE, NAMES, NWORDOUT, WARHEAD, ZONES, 19501
SUBROUTINES CALLED:	EXPF, HIST, HISTWRIT, IFIND, IGET, IPUT, RANF,* VLRAD
CALLED BY:	LATTRIT, TERMBMD

Method

Subroutine HISTWRIT is called to write out the HISTOUT block for the previous event, since BDAMAGE is called directly, and the normal call on HISTWRIT by DONEXT is bypassed. The number and yield of the weapon are cumulated by delivery vehicle class and type. The collocation flag KCOL is initialized to zero (off). Y23 is computed by raising the yield for this weapon type YIELD(NWTYP) to the 2/3 power. The weapon-target coordinates IWTX, IWTY are found by changing the sign of IAGX and IAGY.

In statement 5, function IGET is used with the variable IKEEP of the STATUS array to determine whether the target is dynamic. If not, this is recorded and control transferred to statement 12. Otherwise the status of the target TSTAT is retrieved through a call to IGET and stored in ISTAT. If ISTAT is zero, the target is already dead. This is recorded and control transferred to statement 12. Otherwise, the square (in nautical miles) of the weapon-target radius WTR2 is computed by squaring the distance coordinates and multiplying by a conversion factor. If WTR2 is greater than or equal to WTR21MT, the maximum range of a one-megaton weapon, multiplied by Y23, the target is too distant. This is recorded and control transferred to statement 12.

If the target is within the prescribed distance, the target vulnerability TVULN is retrieved through a call to IGET and stored in IVULN. The

*System Library Function

hardness value for this vulnerability VULN(IVULN) is stored in NVN. Function VLRAD is called to compute the weapon radius against this target, which is stored in WR. The survival probability SSKP is computed. If SSKP is less than or equal to a random number generated by subroutine RANF, the target survives. This is recorded and control transferred to statement 12.

Otherwise, the status of the target is set to "dead" through a call on IPUT and this is recorded. Through calls on IGET, the area defense zone and area defense component index of the target are retrieved from the STATUS array and stored on DEFZONE and DEFCMP.

In either case, the target class is found. If the class is not 4, 5, or 14 control goes to statement 12. If the class is 14 (ABMDEF), control goes to statement 118. Otherwise, JJ is set to the second target type of class IC, and KK is set to the first target type of class IC+1. IT is set to one. The target index INTAR is compared to each INDBEGTY(II), the beginning indices of the types from JJ through KK. If INTAR is less than INDBEGTY(II), IT is incremented by one, and the test is made again. If II exceeds KK, the breakpoint table is in error and an error message is printed. If on the other hand INTAR is greater than or equal to INDBEGTY(II), the target type has been found and ITARTYPE is set to the current value of IT.

When either the target type is found or II exceeds KK, the zone for the target is retrieved through the use of IGET and stored in ZONE. IC is tested.

If IC is four, the target is a command and control site and the command/control potential for the current zone ZCCPOT(IZONE) is decreased by CCPOT(ITARTYPE), the command/control potential of this site. The zone, the zone command/control potential, the target type, and the site command/control potential are then printed. If IC is five, the target is a bomber interceptor base, and the defensive potential for the current zone ZDEFPOT(IZONE) is decreased by DEFPOT(ITARTYPE), the defensive potential of the base. In either case control then goes to statement 12.

In statement 118, if DECMP is three or less, the target is a long-range radar. In this case, the IOVERLAP array is searched by unpacking KTAR in each word of the array and comparing it to INTAR, the current target index. When a match is found in one of the words, the rest of the word, containing the area defense zones over which the radar is effective, is unpacked. For each of these zones NLRR, the number of long-range radars covering an area defense zone, is decreased by one. If NLRR becomes zero in any zone, the loss of radar coverage in that zone is recorded.

If DEFCMP is four, the target is an ABM base and AINT(DEFZONE, DEFCMP), the number of area interceptors at the base, is set to zero. A check is made to see if this reduction of interceptors has exhausted the entire interceptor supply in the zone. If so, this is recorded.

In statement 12, TARDEFHI, the high-altitude defense state for this target, is retrieved through the use of IGET and stored in KDEF. TARDEFLO, the low-altitude defense state for this target, is also retrieved. If LDEF is positive, it is also reduced by one. The new values are set through calls on subroutine IPUT.

KCOL, the collocation flag, is tested. If it is zero, the current target is the primary target of the weapon. ICOL, which indicates if a target is collocated, is retrieved through the use of IGET and stored in KOLOC. If KOLOC is zero, the target is not collocated, and the subroutine exits. If KOLOC is one, INTAR is saved in INTO and KCOL is set to one to indicate that the assessment of a collocated group is beginning. Function IFIND is called to get the index in array COLAR of the information for the current target and to store the index in KIND. The beginning of the collocation assessment is recorded through a call to subroutine HIST. If KIND is zero, the index of the target supposed to be collocated cannot be found in COLAR. This is recorded, and the subroutine exits.

If KCOL is one or if KIND is nonzero, KEYNTA, the number of targets in the collocated group, is retrieved through the use of IGET and stored in NTA. If NTA is nonzero, the next target in the collocated group is located NTA words in COLAR before the current target, so KIND is decreased by NTA. NWORDOUT, the number of words of the HISTOUT block to be used for this event, is set to 19+NTA. Regardless of the value of NTA, KIND is increased by one to point to the next collocated target.

If INTAR equals INTO, the primary target of the weapon has been reached, and the collocation assessment is complete. The subroutine then exits. If they are not equal, KEYDX and KEYDY, the components of the distance from the current target to the next one, are retrieved through the use of IGET and stored, respectively, in IDX and IDY. IWTX and IWTY are respectively increased by IDX and IDY to give the coordinates of the next target. Control is then transferred to statement 5 above.

Subroutine BDAMAGE is illustrated in figure 10.

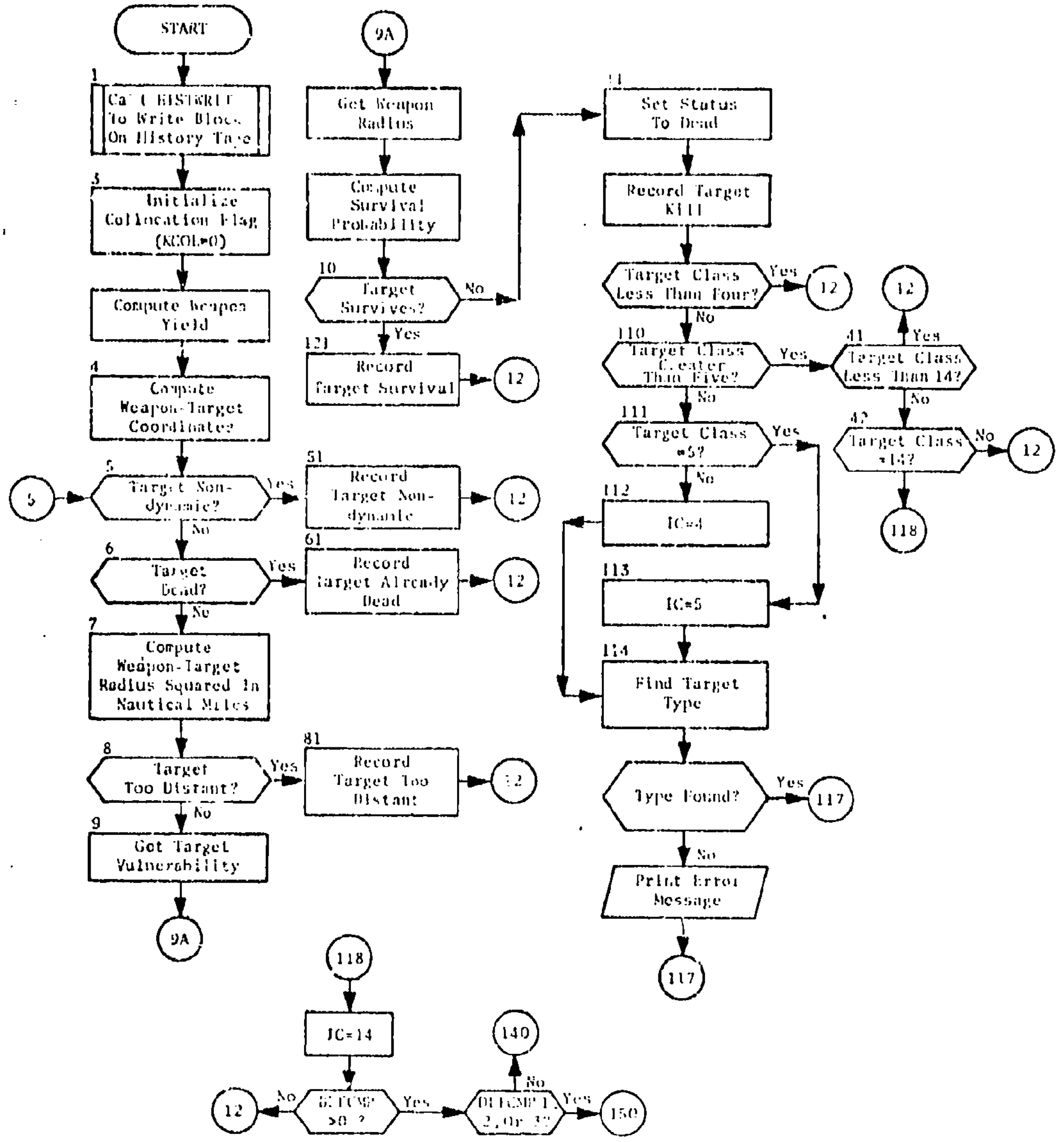


Fig. 10. Subroutine BDAMAGE
(Sheet 1 of 3)

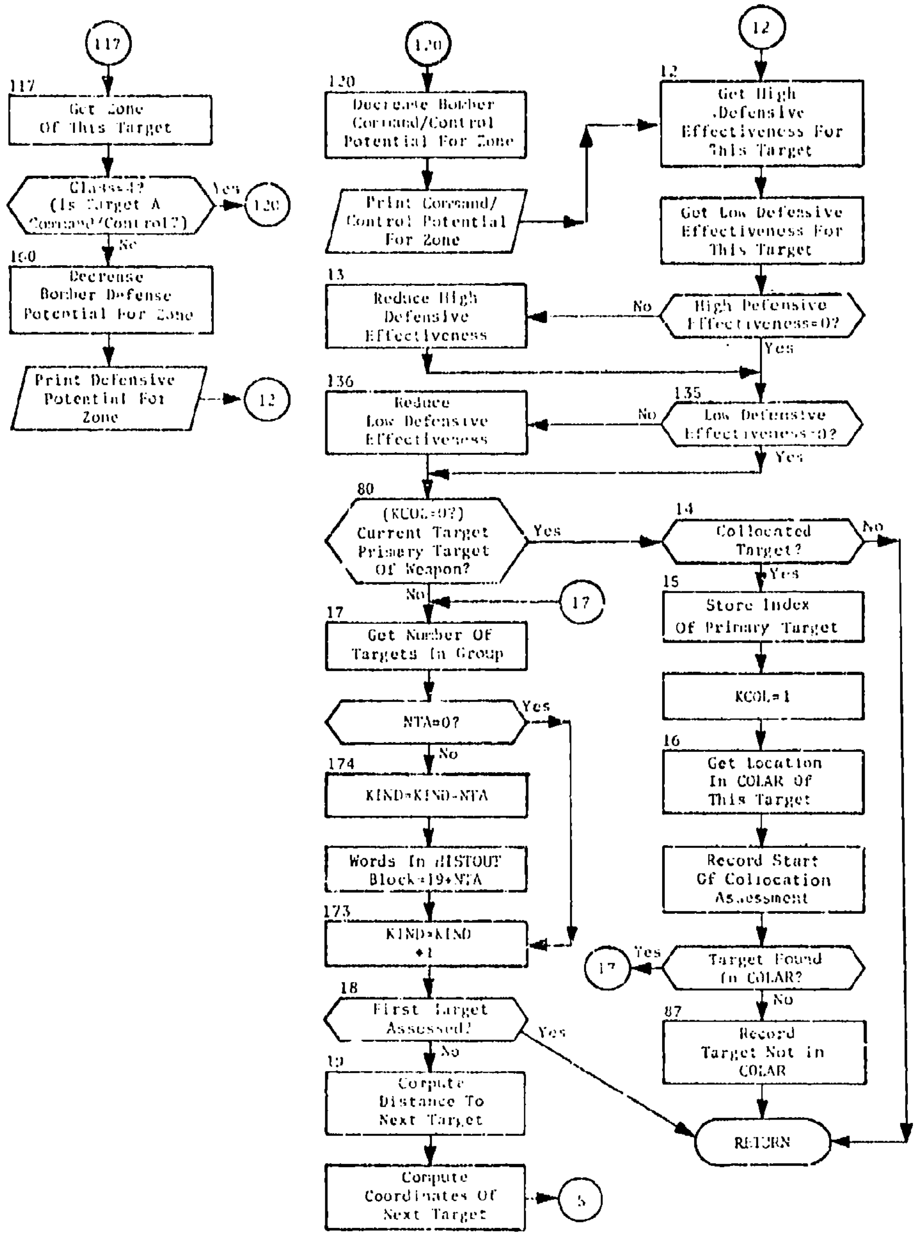


Fig. 10. (cont.)
(Sheet 2 of 3)

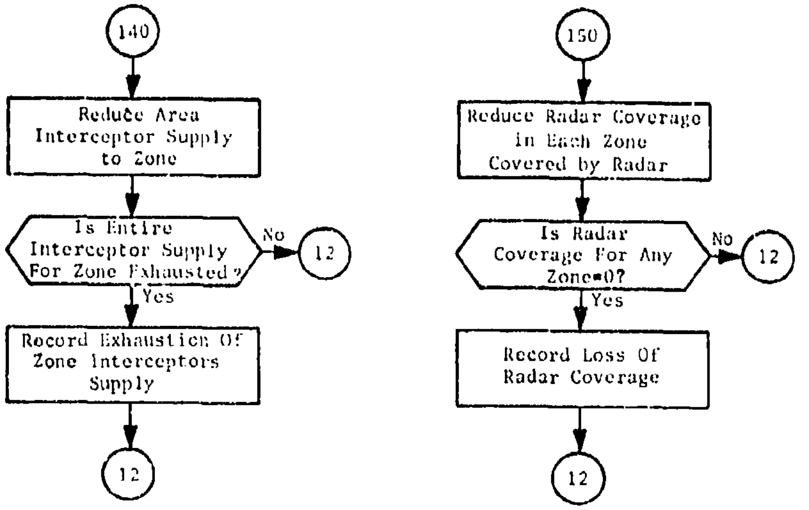


Fig. 10. (cont.)
(Sheet 3 of 3)

SUBROUTINE BLAUN

PURPOSE:	To simulate the attempted launch of a bomber or tanker.
ENTRY POINTS:	BLAUN
FORMAL PARAMETERS:	None
COMMON BLOCKS.	BOMBER, DATTA, EDATA, KEYWORDS, TANKER, TIME, 19501
SUBROUTINES CALLED:	HIST, IGET, LOGF,* NEXTEVNT, PLANTS, RANF*
CALLED BY:	DONEXT

Method

The launch base index INDP(IHT) is retrieved from the History Table and stored in INBASE. The altitude index is set equal to one (high altitude). The execution of BLAUN is recorded through a call to subroutine HIST. The status ISTAT of the launch base is retrieved by use of function IGET.

If the base is dead, this is recorded; no launch occurs, and the subroutine exits.

If the base is alive, the vehicle class ICLASS is tested to determine the vehicle type. If ICLASS is 2, the vehicle is a bomber. If ICLASS is 3, the vehicle is a tanker. In either case, the probability of launch abort PLABT(ITYPE) or TPLABT(ITYPE), delay time TMDEL(ITYPE) or TTMDEL(ITYPE), and abort rate ABRATE(ITYPE) or TABRATE(ITYPE) are stored respectively in PLA, TM, and ABR. PLA is compared to a random number generated by subroutine RANF to determine if abort occurs.

If the vehicle aborts, the vehicle delay time TDEL is computed by multiplying the delay time for this type of vehicle TM by the log of a random number. The launch abort is recorded. A new launch is planted at a time equal to the current game time TIME plus the delay time, and the subroutine exits.

---

*System Library Function

If th vehicle is launched successfully, the inflight abort time TABORT is computed by modifying the abort rate for this type of vehicle ABR by a random factor. The successful launch is recorded. Subroutine NEXTEVNT is called to plant the next bomber event, and the subroutine exits.

Subroutine BLAUN is illustrated in figure 11.

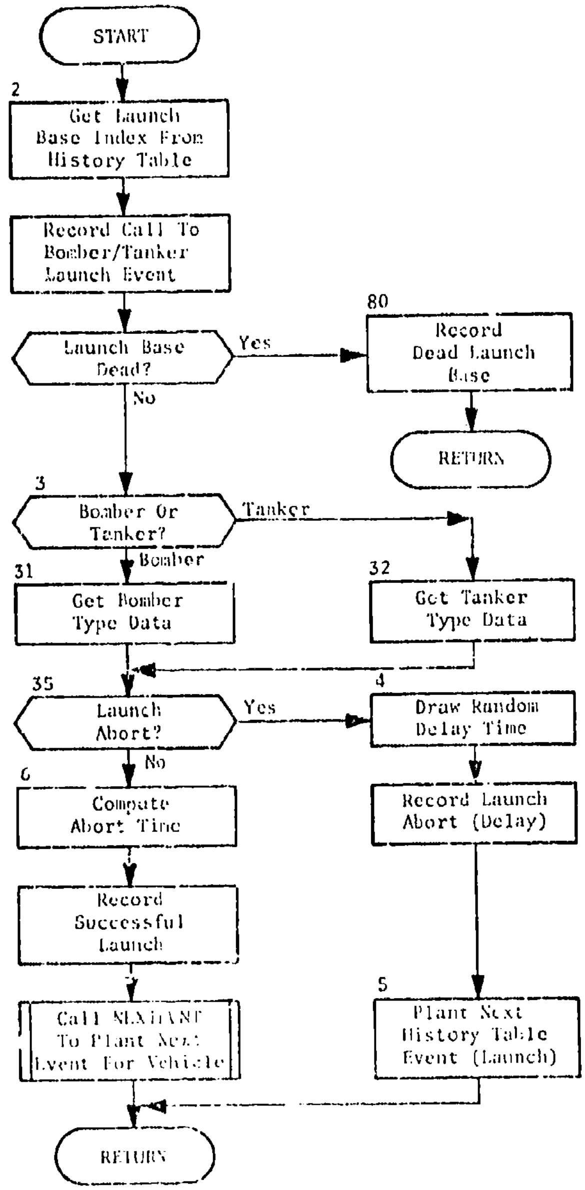


Fig. [illegible]. Subroutine BLAUN

SUBROUTINE BMDSTAT

PURPOSE: To summarize the number of remaining area and terminal interceptors.

ENTRY POINTS: BMDSTAT

FORMAL PARAMETERS: None

COMMON BLOCKS: ABMDATA, AREADAT, BRKPNT, KEYWORDS, NAMES, TBMDATA, 19501

SUBROUTINES CALLED: None

CALLED BY: SIMULATE, ENDGAME

Method

A printout is made of the number of area interceptors for each side that are present in each zone and of the number of terminal interceptors present at each target which is protected by a local ballistic missile defense.

Subroutine BMDSTAT is illustrated in figure 12.

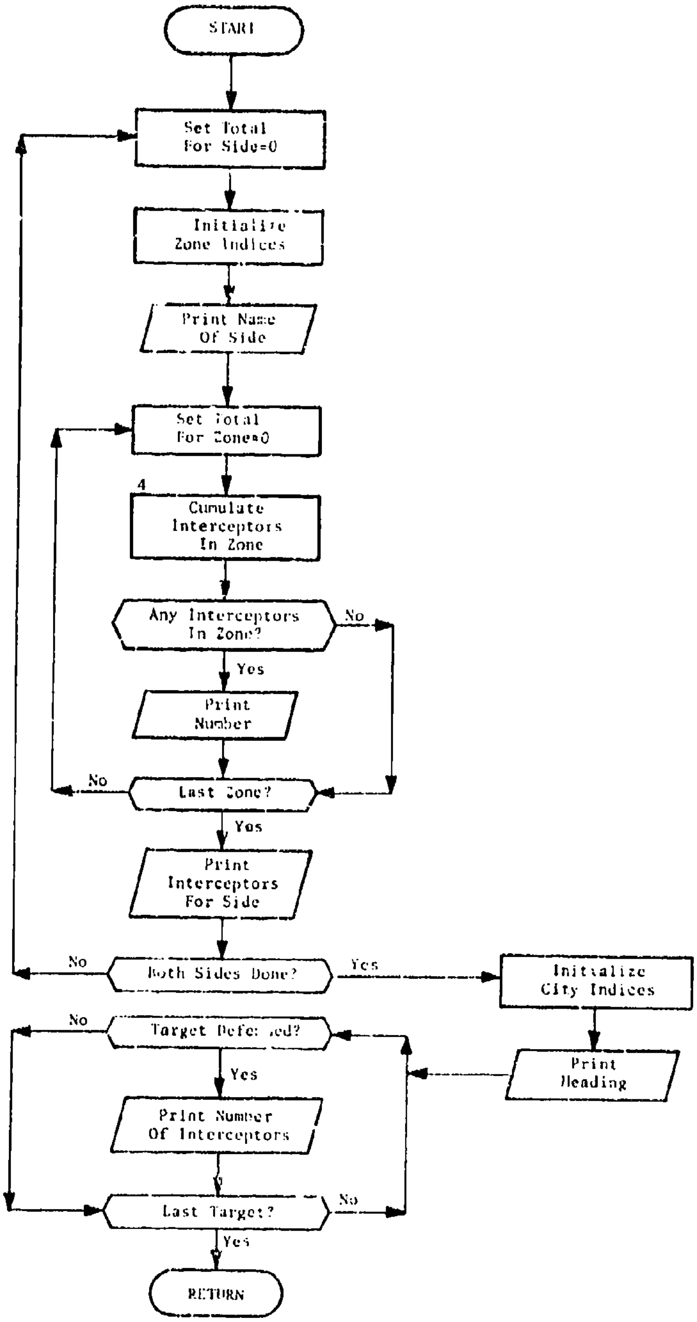


Fig. 12. Subroutine BDMSTAT

SUBROUTINE BOMBF

PURPOSE:	To store in the HISTOUT block the unused History Table and Weapon Table lines of a killed bomber.
ENTRY POINTS:	BOMBF
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, HISTOUT, NWORDOUT, INBOMBF
SUBROUTINES CALLED:	MINOF *
CALLED BY:	HIST

Method

If the variable INBOMBF has been set equal to zero by card input, NWORDOUT (the number of words of the HISTOUT block to be output) is set to 25, and the subroutine exits. Otherwise, the number of lines remaining in the History Table, NMHT, is calculated by subtracting NALT, the number of alternate events for this vehicle, from JHF. The unused History Table elements: TINC(MHT-I+1) time increments; INDP(MHT-I+1) places; and INDE(MHT-I+1) event types; are stored respectively in TINCN(I), INDPN(I), and INDEN(I).

The type designation for a maximum of six unused weapons (weapons not delivered prior to bomber kill) are also stored. The system function MINOF is used to compare the number six with the number of unused weapons and return the minimum value J. The data for the unused weapons, up to six, in the Weapon Table IWTYP(JWT-I+1) are then stored in NWTYPEN(I). NWORDOUT is set to 152 and the subroutine exits.

Subroutine BOMBF is illustrated in figure 13.

---

*System Library Function

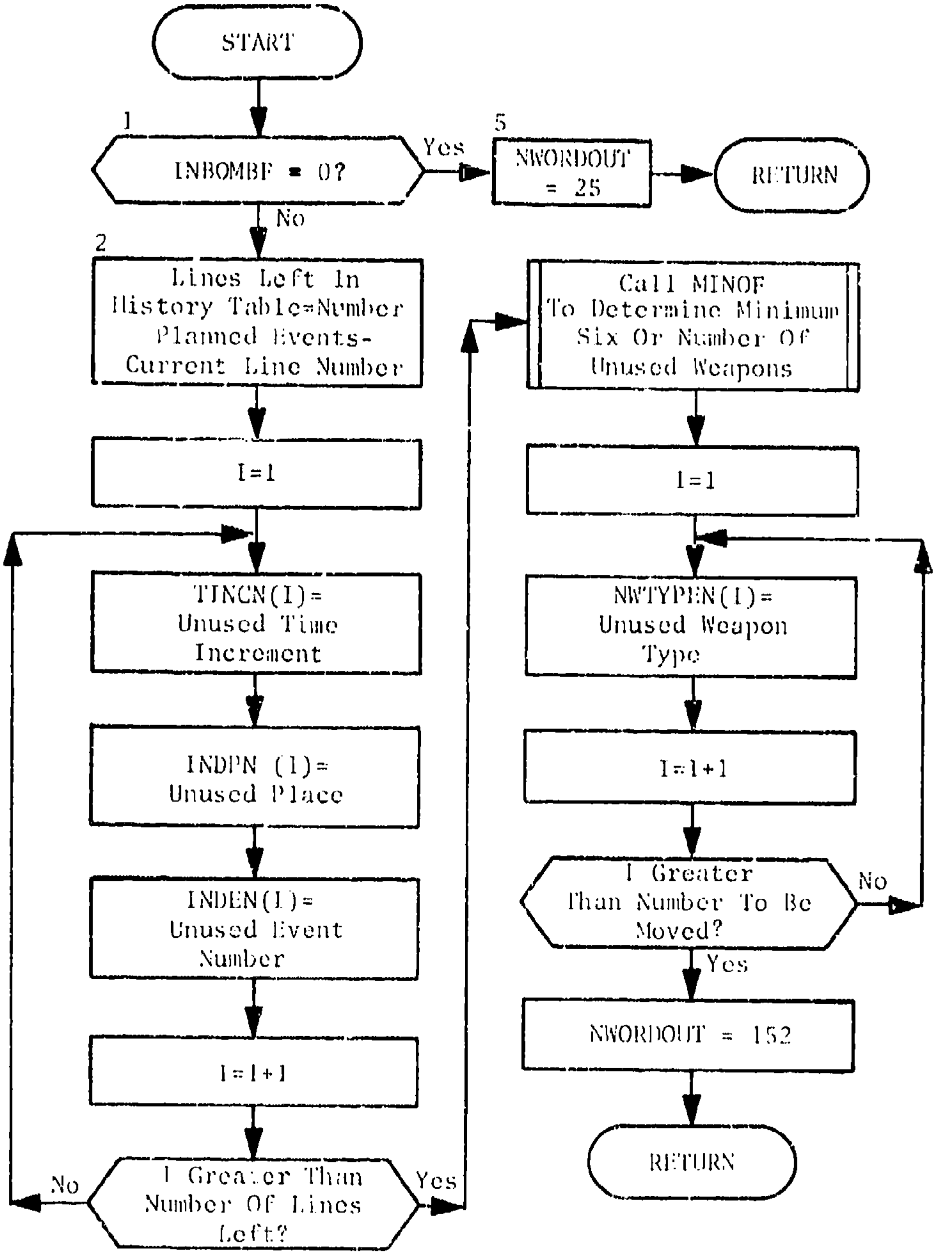


Fig. 13. Subroutine BOMBF

SUBROUTINE BTINPRIN

PURPOSE:	To print out the contents of the INDATA array for a bomber or tanker sortie.
ENTRY POINTS:	BTINPRIN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, IPSWICH
SUBROUTINES CALLED:	None
CALLED BY:	SIMULATE

Method

This subroutine is called by program SIMULATE only if IPSW2 (for bombers) or IPSW3 (for tankers) has been set equal to one by card input. The first 26 words of INDATA are printed; then the time, place, and event types for the events listed in the History Table are printed out in groups of 10. If a bomber and IPSW6 have been set to one, the warhead types and desired coordinates are also printed out.

Subroutine BTINPRIN is illustrated in figure 14.

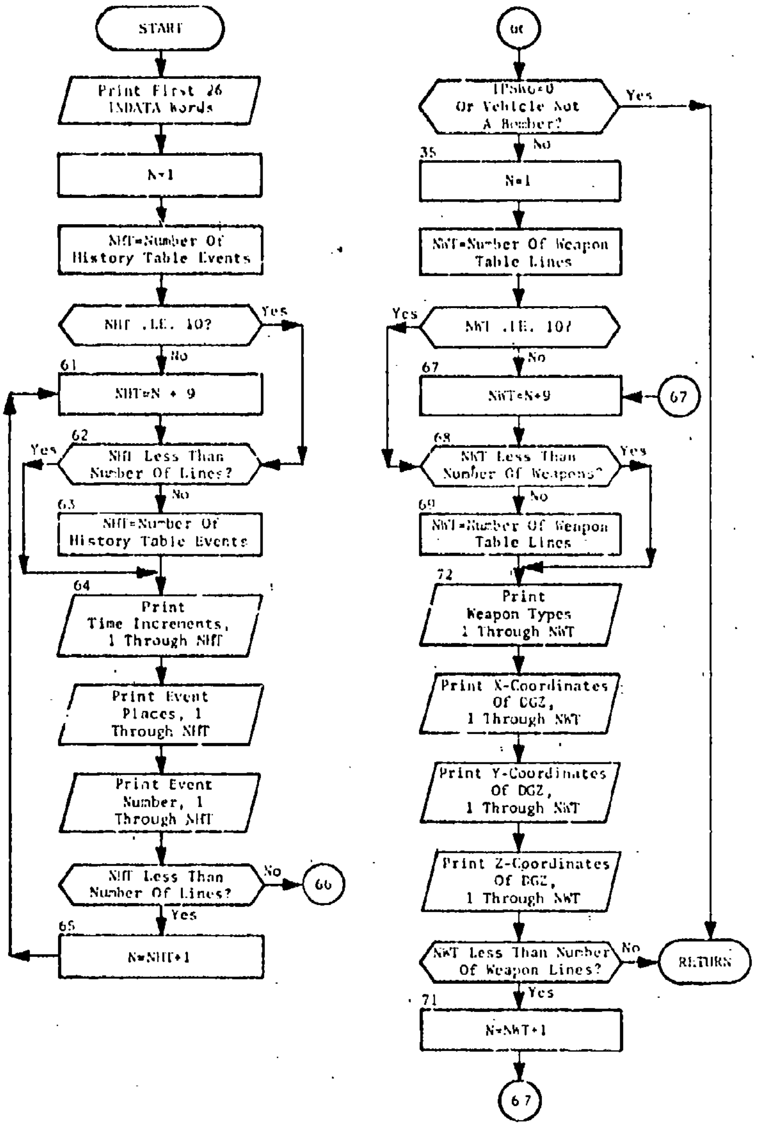


Fig. 14. Subroutine BTINPRIN

SUBROUTINE CHANGALT

PURPOSE:	To change a bomber's altitude index.
ENTRY POINTS:	CHANGALT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA
SUBROUTINES CALLED:	HIST, NEXTEVNT
CALLED BY:	DONEXT

Method

The execution of subroutine CHANGALT is recorded through a call to. subroutine HIST.

The altitude index IALT is set to zero if it was one, or to one if it was zero. (Zero is low altitude, one is high altitude.)

Subroutine NEXTEVNT is called to plant the next event for the bomber.

Subroutine CHANGALT is illustrated in figure 15.

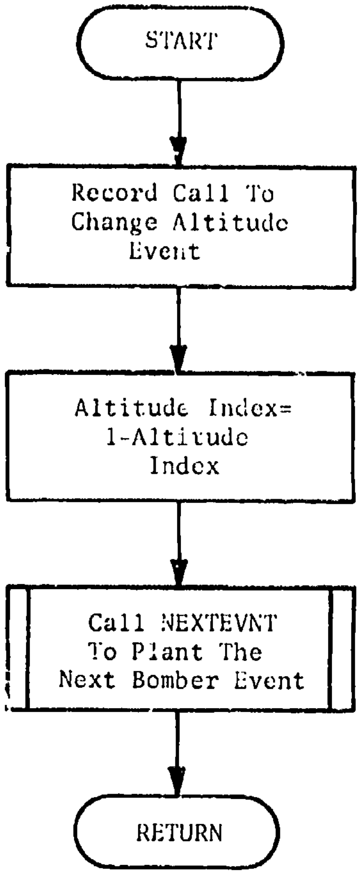


Fig. 15. Subroutine CHANGALT

SUBROUTINE CLAUN

PURPOSE:	To test if a launched missile successfully completes powered flight, and to plant an Area Ballistic Missile Defense Attrition event for each MIRV for the end of its flight time.
ENTRY POINTS:	CLAUN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ABMDATA, EDATA, KEYWORDS, MISSLE, PAYLOAD, TIME, 19S01
SUBROUTINES CALLED:	HIST, IGET, IPUT, PLANTS, MUNPK
CALLED BY:	DONEXT

Method

The launch base is restored to its initial hardness by a call to IPUT; the base status is then retrieved and tested. If the launch base is dead, the destruction of the missile during launch phase is recorded and the subroutine exits.

If the launch base is still alive, the payload data are retrieved and, if the payload contains several MIRVs, that fact is recorded. The time over target, the DGZ, and the decoy data for each warhead are used to plant AREABMD events.

Subroutine CLAUN is illustrated in figure 16.

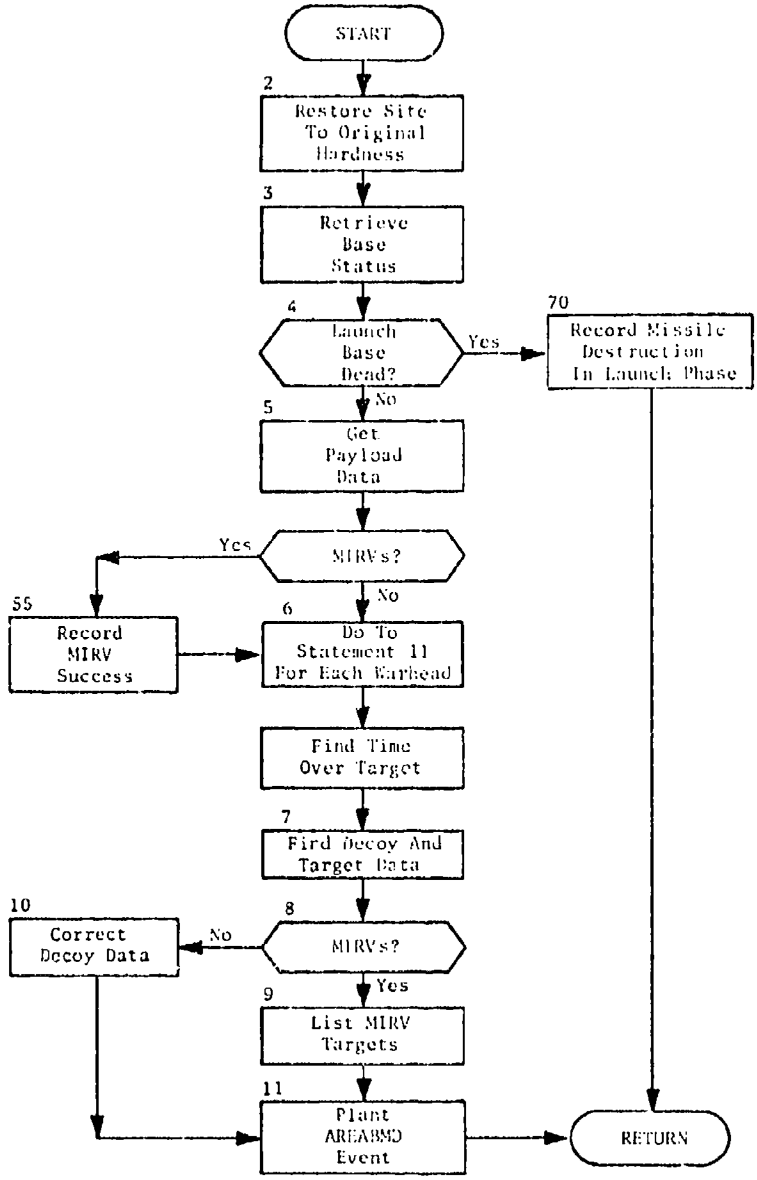


Fig. 16. Subroutine CLAUN

SUBROUTINE DLAUN

PURPOSE: To launch or terminate decoys from a bomber, at high or low altitude.

ENTRY POINTS: DLAUN

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA, TIME, ZONES

SUBROUTINES CALLED: HIST, NEXTEVNT

CALLED BY: DONEXT

Method

The number of penetrators in the zone is changed by the amount specified in the "Place" portion of the bomber History Table. If the change is negative, the number of high-altitude decoys accompanying the bomber is decreased accordingly. If the change is positive, either the high-altitude or the low-altitude decoys are incremented, depending upon the current bomber altitude. The execution of DLAUN is recorded by calls on subroutine HIST. Subroutine NEXTEVNT is then called.

Subroutine DLAUN is illustrated in figure 17.

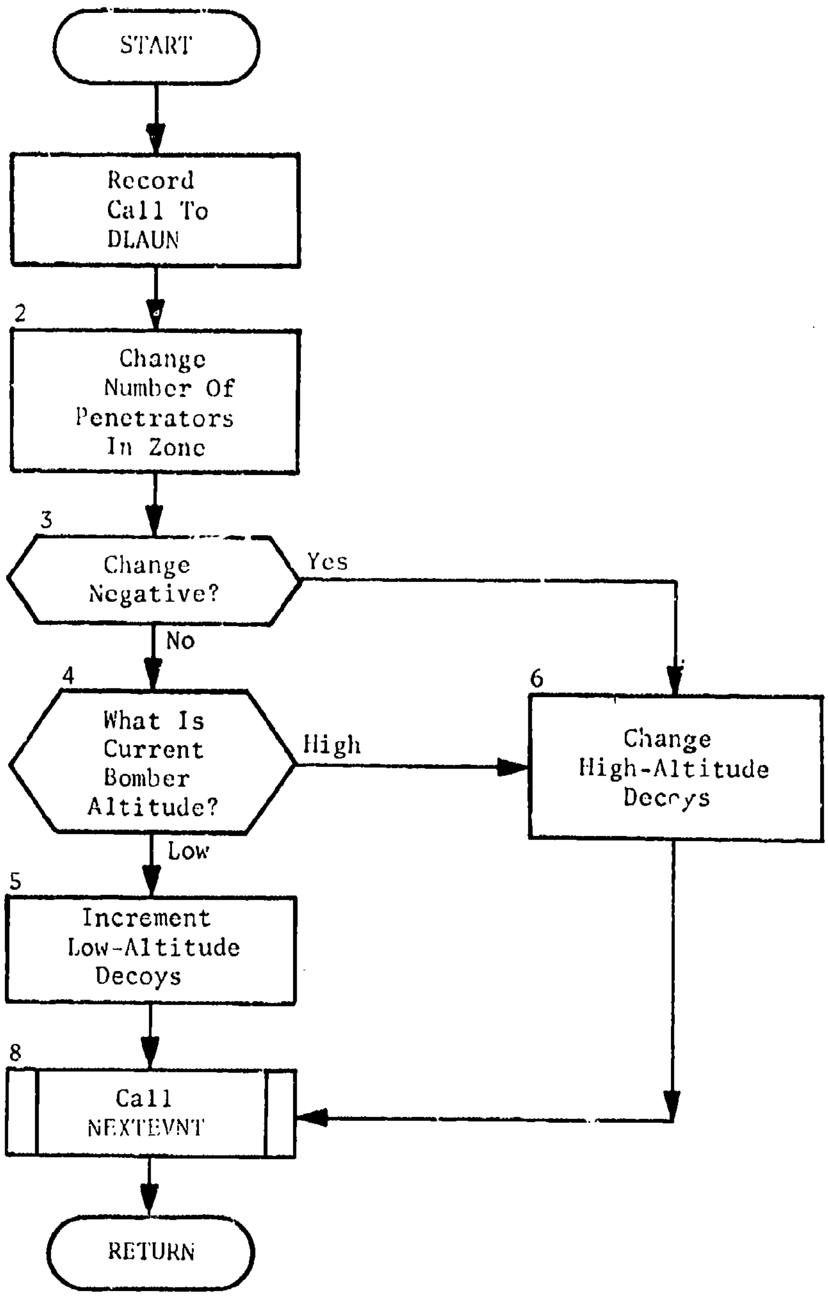


Fig. 17. Subroutine DLAUN

SUBROUTINE DONEXT

PURPOSE:	To determine the next event to be executed, either from internal memory or from an external spill file, and to call the appropriate event subroutine. Entry point EVSPILL empties the list memory onto an external spill file when list memory is filled.*
ENTRY POINTS:	DONEXT, EVSPILL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CONST, ESTOR, EVENT, EVINDX, ITP, MONDAT, MYIDENT, NEVTOT, TIME, TWORD, 19501
SUBROUTINES CALLED:	AATTRIT, ALAUN, AREABMD, BABORT, BDAMAGE, BLAUN, CHANGALT, CLAUN, DLAUN, ENDGAME, ERAREA, ESEC, EVUNPK, HISTWRIT, LATTRIT, LRAREA, MLAUN, MONPRIN, NAVATR, NAVCAL, RDARRAY, RDWORD, RECHEK, RECOVERY, REFUEL, RETLM, SETREAD, SETWRITE, SQUEEZE, SSTAT, TERMBMD, TERMTAPE, UPDIR, WRARRAY, WRWORD, ZABORT
CALLED BY:	SIMULATE

Method

Subroutine DONEXT controls the program and never exits once it is called.

When executed, DONEXT determines if the next event comes from list memory or from one of the event tapes. If the next event to be executed is in list memory, the event is transferred from list memory to the array INDATP. The memory cells used by the event are returned to available storage; the data in INDATP are unpacked and placed in the array INDATA. The appropriate event subroutine is then called. After the event routine has been executed, the results are written on the output tape by subroutine HISTOUT. DONEXT then goes back to the beginning to find the next event. If the next event is on tape, the event data are read directly into INDATP and executed as previously described.

*For discussion of list memory, see Concept of Operation, this chapter.

Entry EVSPILL

The EVSPILL entry point is used, when list memory is full, to merge memory to tape. The array ITOGO contains the number of words written on each of the event spill tapes. Upon entry, EVSPILL selects the tape with the minimum number of words and uses this as an input tape and the first empty tape as the output tape. The merge switch MERGSW is set negative, and control is transferred to the beginning of DONEXT, where an event is selected from LIST memory or the selected input tape and transferred to INDATP. When the merge switch is negative, the selected event will be written on the output tape instead of executed. When the merge is completed, the output tape is put into read status, the number of words on the input tape is set equal to zero, and the arrays that were saved are restored.

Subroutine DONEXT is illustrated in figure 18.

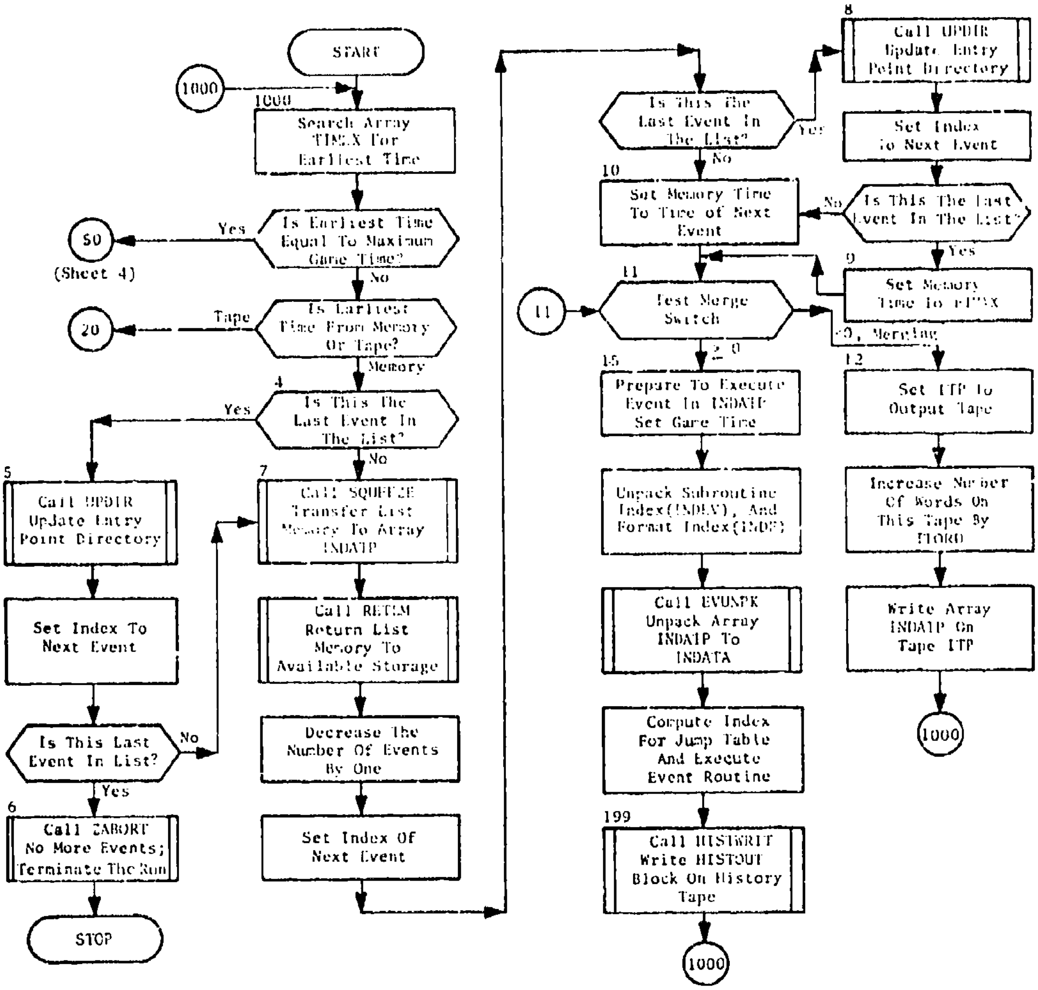


Fig. 18. Subroutine DONEXT
(Sheet 1 of 4)

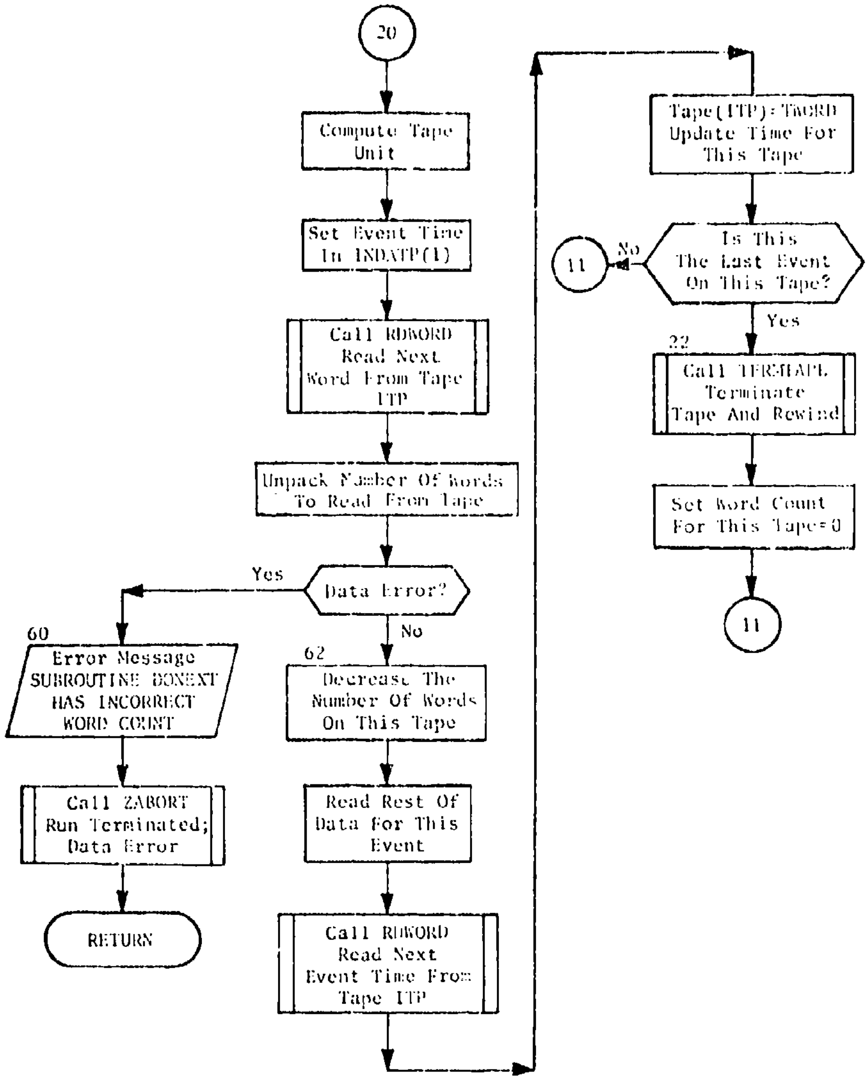


Fig. 18. (cont.)
(Sheet 2 of 4)

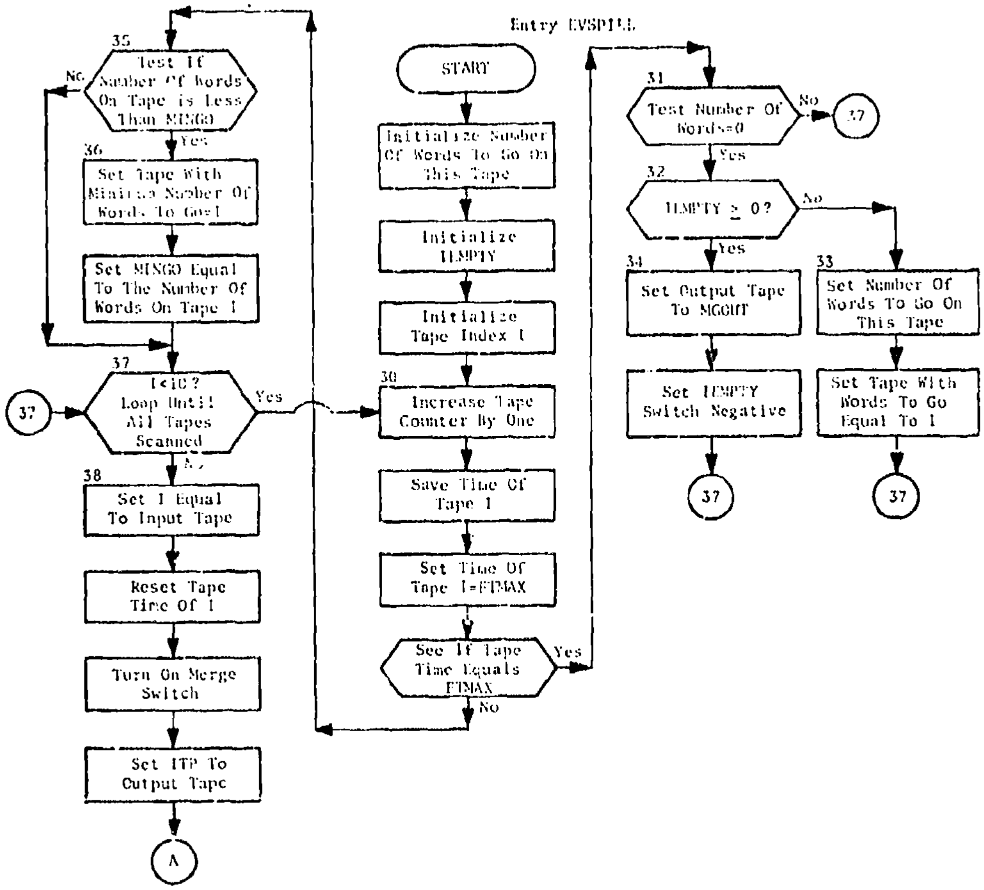


Fig. 18. (cont.)
(Sheet 3 of 4)

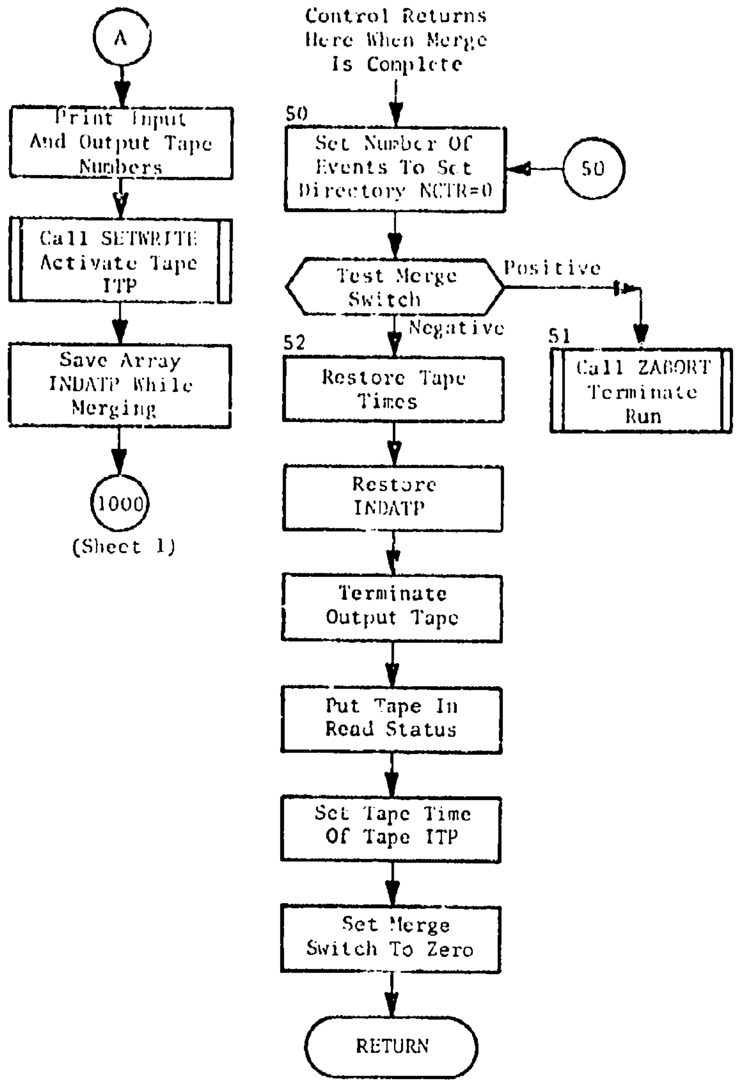


Fig. 18. (cont.)
(Sheet 4 of 4)

SUBROUTINE ENDGAME

PURPOSE:	To perform various summary prints and prepare for termination of the program.
ENTRY POINTS:	ENDGAME
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BRKPNT, ITP, MYIDENT, NAMES, RECOV, TIME, TWORD, WARHEAD
SUBROUTINES CALLED:	AGZSUM, BMDSTAT, PAGESKP, SETREAD, SKIPFILE, SSTAT, STATSUM, TERMTAPE, WRARRAY, WRWORD
CALLED BY:	DONEXT

Method

Summary information is printed out either directly or by calls on appropriate subroutines. The word 4HLAST is written on the History tape to indicate to the Data Output subsystem that there are no more events. The recovery arrays are then written out. After terminating the tape with TERMTAPE, which rewinds the tape, the tape is read down to the end of file, and the breakpoint tables and yield tables are written on the tape without the use of the filehandler. The subroutine then indicates program termination on the typewriter and stops.

Subroutine ENDGAME is illustrated in figure 19.

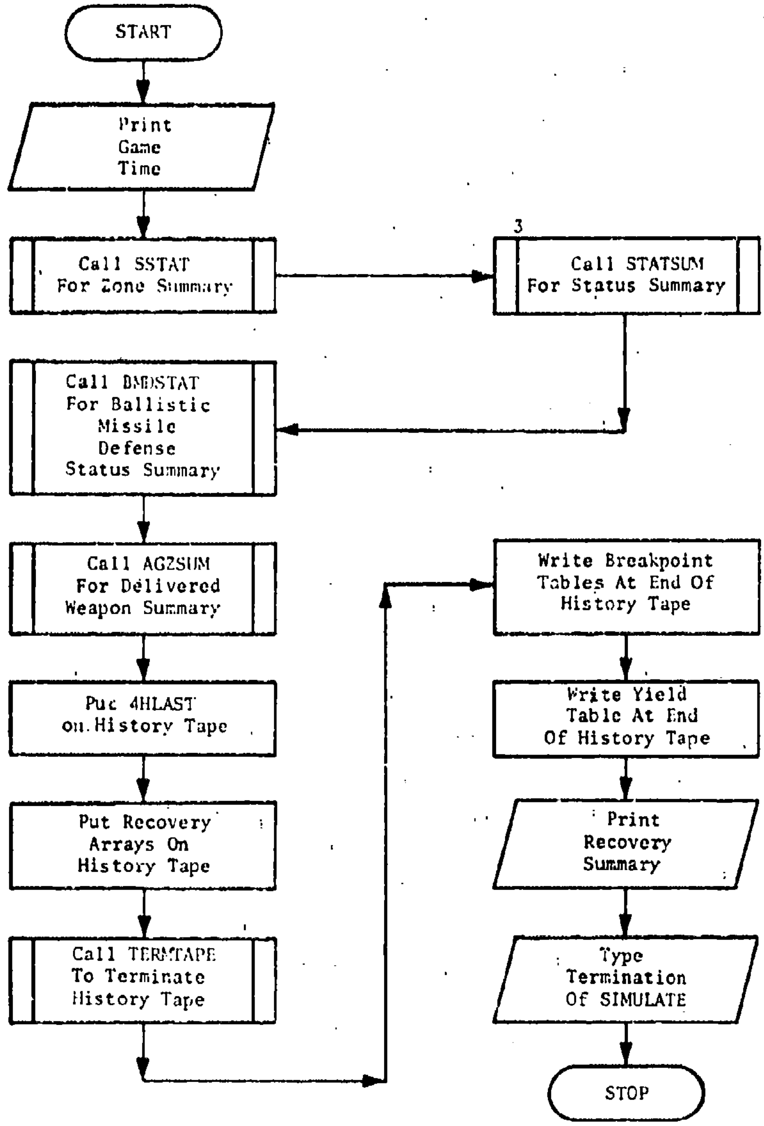


Fig. 19. Subroutine ENDGAME

SUBROUTINE ERAREA

PURPOSE:	To simulate the arrival of a full tanker at a refueling area.
ENTRY POINTS:	ERAREA
FORMAL PARAMETERS:	None
COMMON BLOCKS:	DATTA, EDATA, EPSN, REFUEL, TIME
SUBROUTINES CALLED:	HIST, PLANTS
CALLED BY:	DONEXT

Method

The execution of ERAREA is recorded through a call to subroutine HIST. The index of the refueling area INDRA is retrieved from the tanker History Table and the amount of fuel in the area NFTANK(INDRA) is increased by 60 units.

The History Table pointer IHT is advanced to the next event, and the tanker abort time TABORT is examined to see if it is earlier than FUTIME, the time of the next tanker History Table event. A tanker abort is interpreted as an early departure from the refueling area. If an abort occurs, FUTIME is set to the abort time.

In either case, a Leave Refuel Area event is planted for the tanker, and the subroutine exits.

Subroutine ERAREA is illustrated in figure 20.

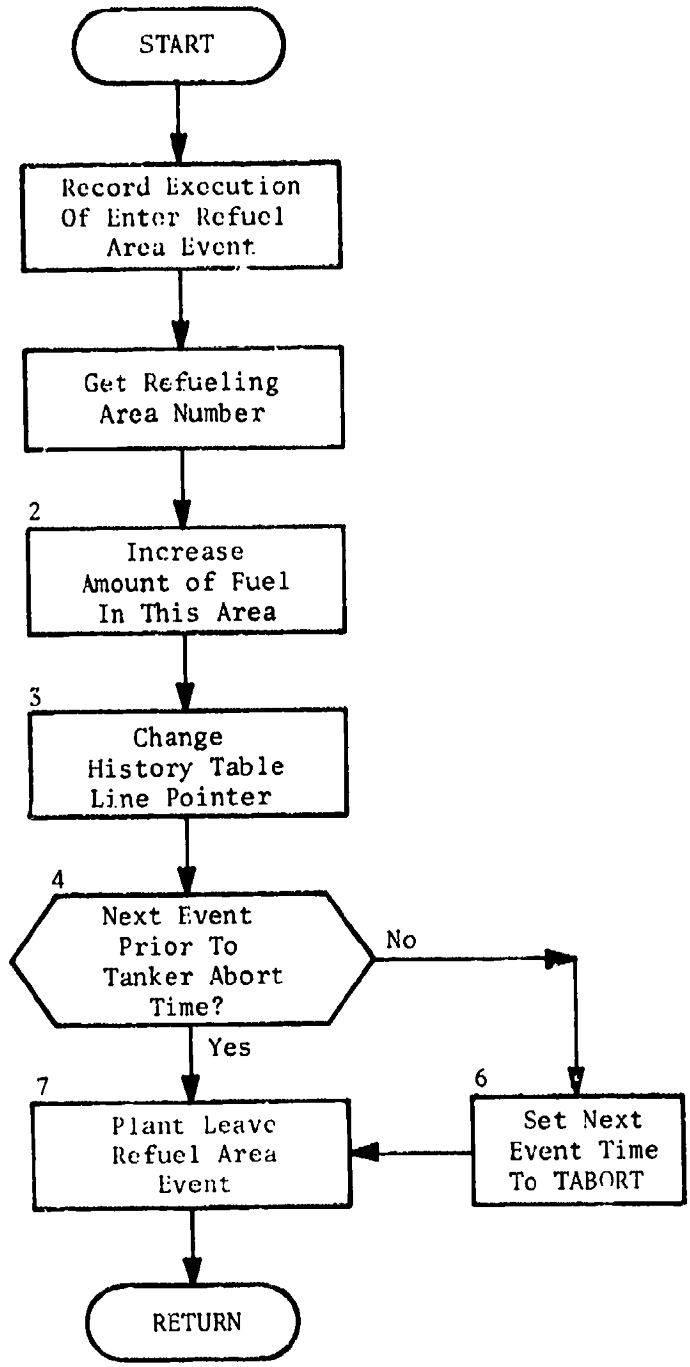


Fig. 20. Subroutine ERAREA

SUBROUTINE ERROL

PURPOSE:	To print an error message for subroutine PLANT or subroutine UNSQUEEZ and call subroutine ZABORT to terminate the game.
ENTRY POINTS:	ERROL
FORMAL PARAMETERS:	ITYP - Error type
COMMON BLOCKS:	None
SUBROUTINES CALLED:	ZABORT
CALLED BY:	PLANT, UNSQUEEZ

Method

When called, this subroutine prints the error message PLANT OR UNSQUEEZ ERRORxx, where xx indicates one of the error types described below. ERROL then calls subroutine ZABORT to terminate the run. Subroutine ERROL is illustrated in figure 21.

ERROR TYPE	DESCRIPTION
10, 11	No list memory available after EVSPILL. Called by subroutine PLANT.
20	No list memory available after EVSPILL. Called by subroutine UNSQUEEZ.
21	Event execution time earlier than game time. Called by subroutine PLANT.

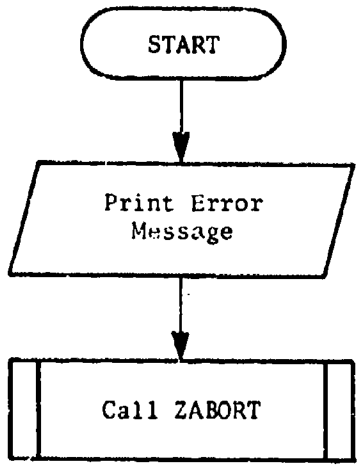


Fig. 21. Subroutine ERROL

SUBROUTINE ESEC

PURPOSE:	To adjust the number of penetrators in the old and new zones as a bomber enters a zone; if this is a depenetration from enemy territory, to record this and to determine the proper recovery base; and to plant the next event for the bomber.
ENTRY POINTS:	ESEC
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, EPSN, HISREC, HISTOUT, KEYWORDS, RECOV, TIME, ZONES, 19501
SUBROUTINES CALLED:	HIST, NEXTEVNT, PLANTS
CALLED BY:	DONEXT

Method

The execution of ESEC is recorded through a call to subroutine HIST. If the current zone INZONE is not the bomber's launch zone, the number of penetrators in that zone NPENZ(INZONE) is decreased by one plus the number of decoys accompanying the bomber. If it is the bomber launch zone, the entry of enemy territory is recorded. In any case, the new zone INDP(IHT) is retrieved from the History Table and the zone entry recorded.

The number of penetrators in the new zone is increased by one for the bomber and by the number of its accompanying decoys, if any. If the zone crossing is internal, the History tape record is suppressed.

If the bomber is leaving enemy territory, the depenetration corridor is determined and the depenetration recorded. The furthest live base which is not saturated is then selected for recovery, if one is available. If there is no such base available, either a saturated base is selected or the bomber is aborted.

Subroutine ESEC is illustrated in figure 22.

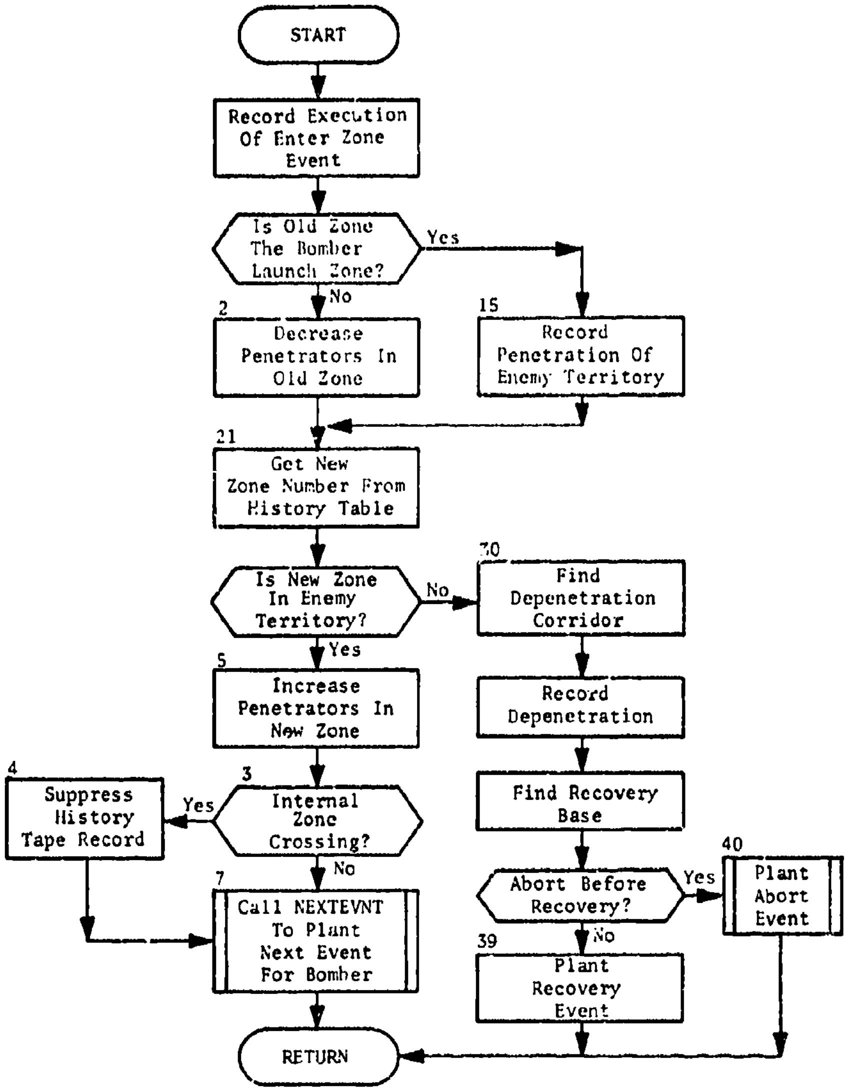


Fig. 22. Subroutine ESEC

SUBROUTINE EVPACK

PURPOSE:	To pack the array OUTDATA into the array OUTDATAP.
ENTRY POINTS:	EVPACK
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, EPACK, ESTOR, FORMAT, MONDAT
SUBROUTINES CALLED:	PROUTDAT, UNPFOR, ZABORT
CALLED BY:	PLANT

Method

A word in the array JFORMAT specifies how to transfer a word or array from IOUTD (equivalent to OUTDATA) to IOUTDP (equivalent to OUTDATAP). If the JFORMAT word has the value zero, the IOUTD word need not be transferred. If it has the value one, the IOUTD word is transferred directly. If it has a value greater than one, an array is to be transferred and the last 12 bits specify the dimension of the array. The remaining bits give the index to the array IOUTD which contains the number of words in the array to be transferred. To transfer all the variables in IOUTD requires the specification of a number of JFORMAT words.

Subroutine EVPACK first calls subroutine UNPFOR, which places the set of indices to the array JFORMAT into the array INDFORMO. The words in INDFORMO are then looked at one at a time, the corresponding JFORMAT determined, and data transferred to OUTDATAP accordingly.

After transferring the data, a check is made, and if more than the allowed number have been transferred, the run aborts.

If MONSW is set equal to one when EVPACK is called, the arrays OUTDATA and OUTDATAP are printed out.

Subroutine EVPACK is illustrated in figure 23.

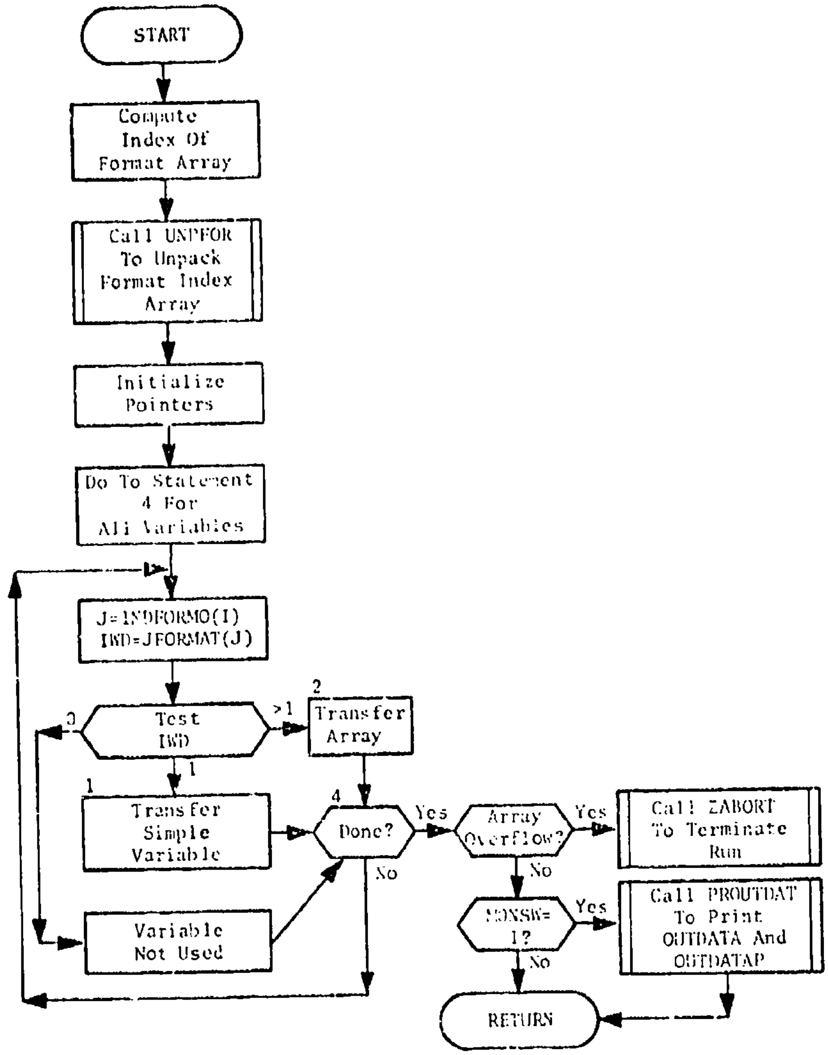


Fig. 23. Subroutine EVPACK

SUBROUTINE EVUNPK

PURPOSE:	To transfer the array INDATAP to the array INDATA for use by an event subroutine.
ENTRY POINTS:	EVUNPK
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, EPACK, ESTOR, FORMAT, MONDAT
SUBROUTINES CALLED:	PRINDAT, UNPFOR, ZABORT
CALLED BY:	DONEXT

Method

The operation of EVUNPK is very similar to that of EVPACK. A call to UNPFOR puts the indices to the array JFORMAT in array INDFORMD. These format specifications are then used to transfer single words and arrays from INDATAP to INDATA.

If more data are transferred than INDATA is dimensioned for, the run aborts.

If MONSW is equal to one when EVUNPK is called, PRINDAT is called to print out INDATA and INDATAP.

Subroutine EVUNPK is illustrated in figure 24.

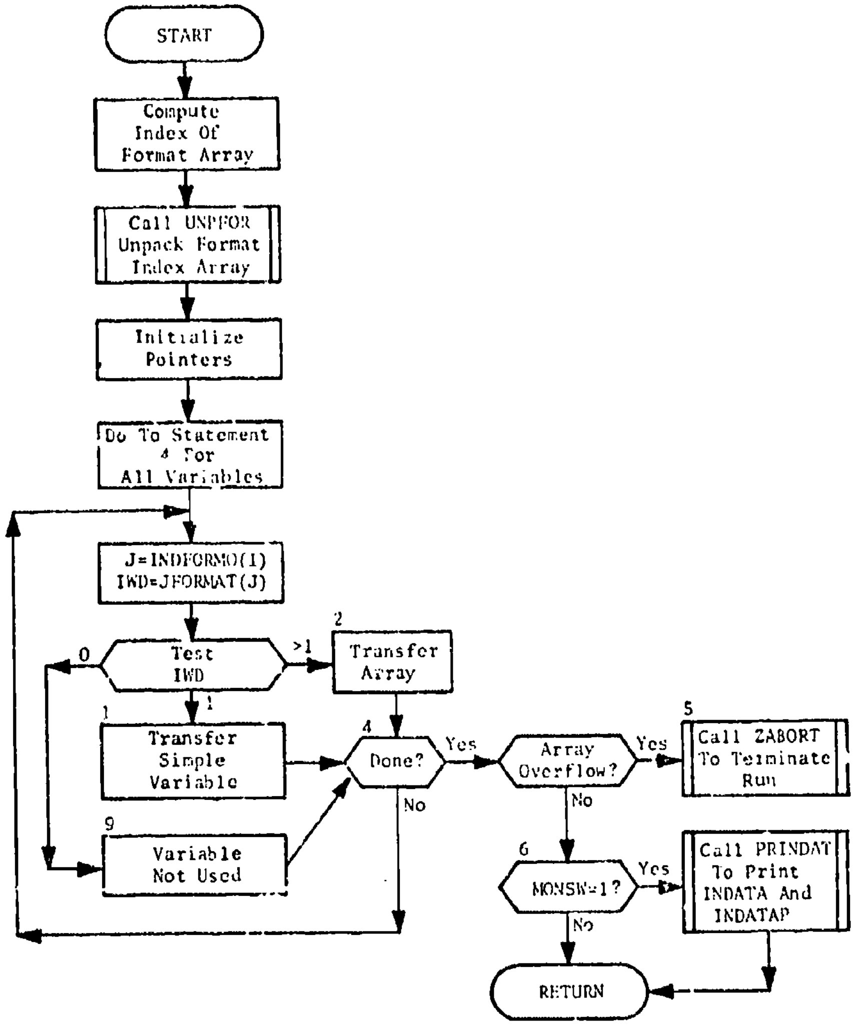


Fig. 24. Subroutine EVUNPK

SUBROUTINE HIST

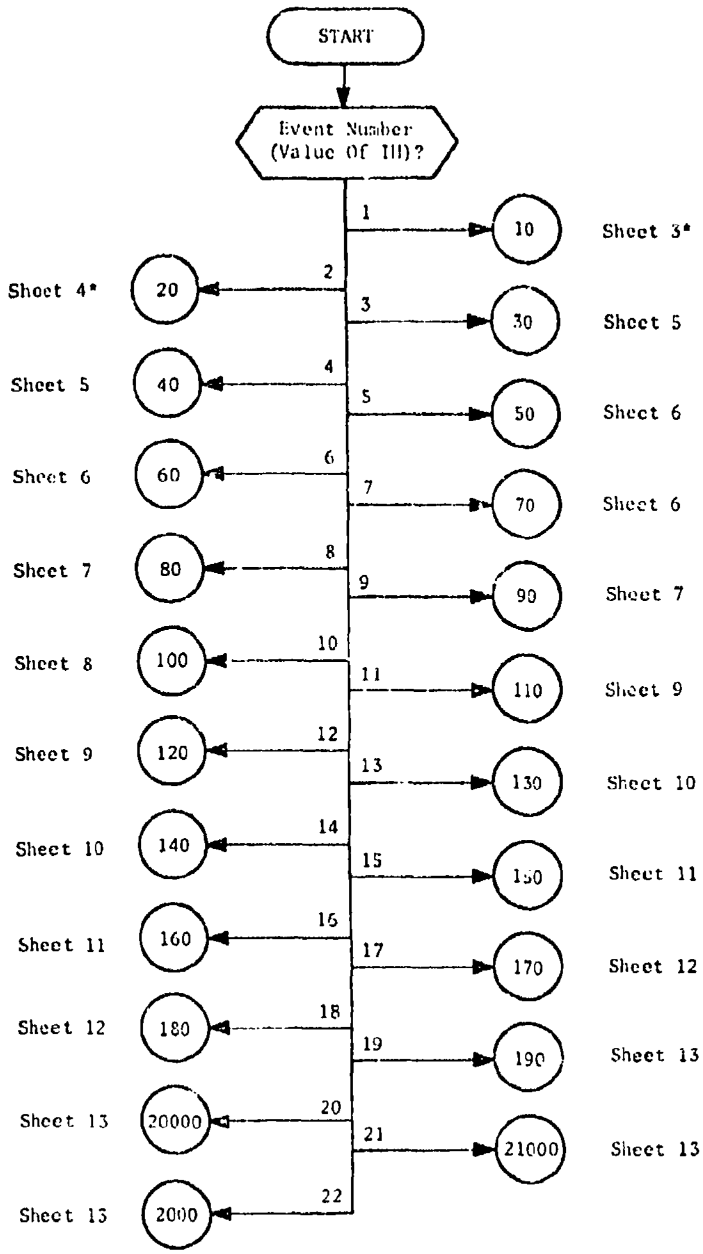
PURPOSE:	To record the outcome of each event on the History tape. Prints of the occurrences are optional.
ENTRY POINTS:	HIST
FORMAL PARAMETERS:	IH - Event number JH - Message number for the event
COMMON BLOCKS:	AREADAT, BOMBER, BRKPNT, DEFZONE, EDATA, GROUND, HISREC, HISTA1, HISTABM, HISTOUT, HISTREF, IPRINT, KEYWORDS, MISSLE, NAMES, NWORDOUT, TIME, TRYL, WARHEAD
SUBROUTINES CALLED:	BOMBF
CALLED BY:	AATTRIT, ALAUN, AREABMD, BABORT, BDAMAGE, BLAUN, CHANGALT, CLAUN, DLAUN, ERAREA, ESEC, LATTRIT, LRAREA, MLAUN, RECHEK, RECOVERY, REFUEL, TERMBMD, TRYLAUN

Method

Based on the value of IH, the event number, control is transferred to a statement number which has been assigned a number equal to 10 times the value of IH. Each of these statements is also a computed GO TO statement.

Based on the value of JH, the message number, control is transferred to the appropriate message statement corresponding to the message of this number for the current event. At each message statement, the appropriate /HISTOUT/ and other common block items for this event and message are set, a message may be printed, and the subroutine exits. Optional prints are controlled by common block /IPRINT/.

Subroutine HIST is illustrated in figure 25.



* "Sheet 3" implies "Sheet 3 of 13."

Fig. 25. Subroutine HIST
(Sheet 1 of 13)

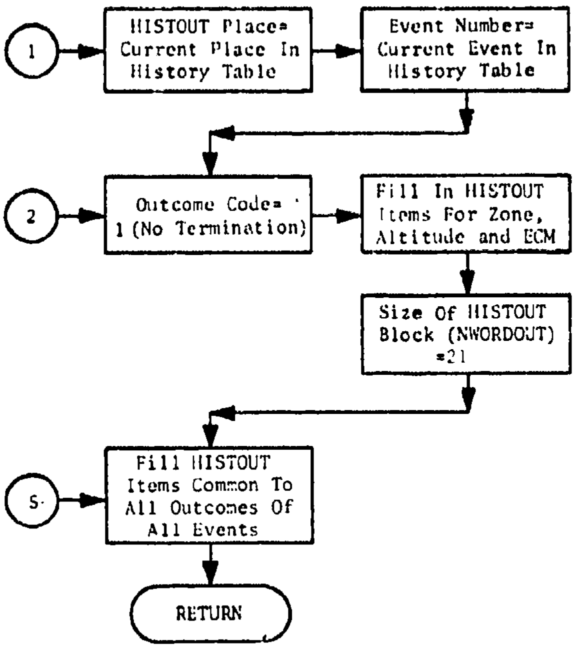


Fig. 25. (cont.)
(Sheet 2 of 13)

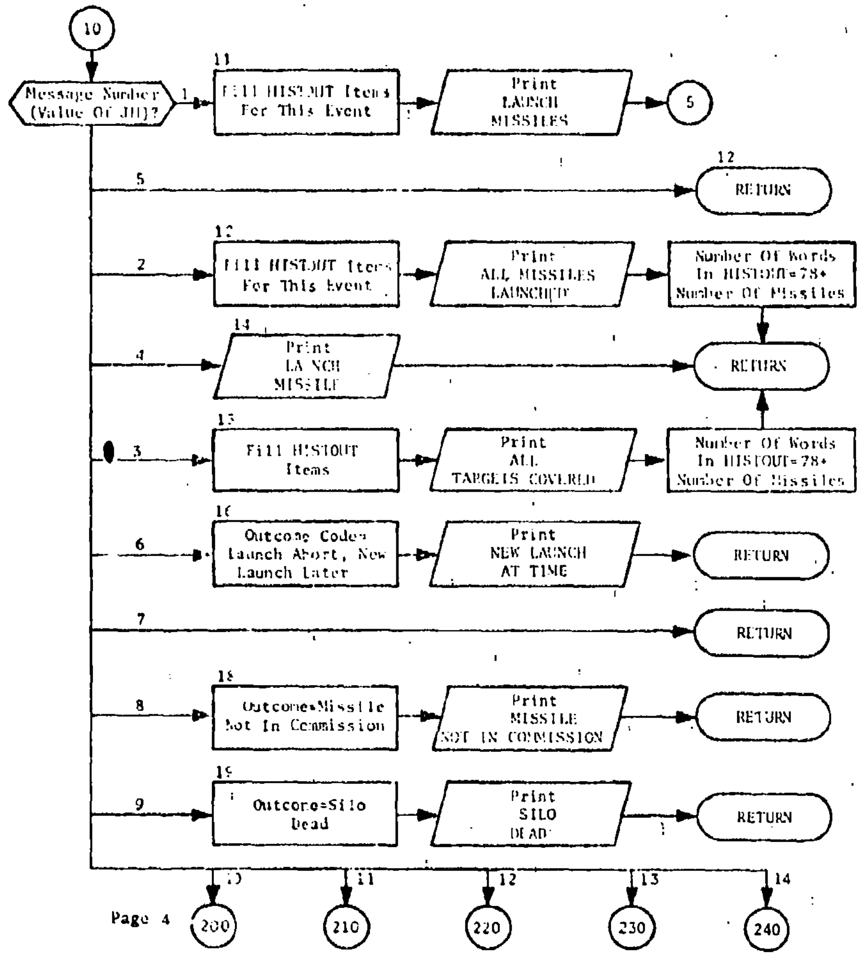


Fig. 25. (cont.)
(Sheet 3 of 13)

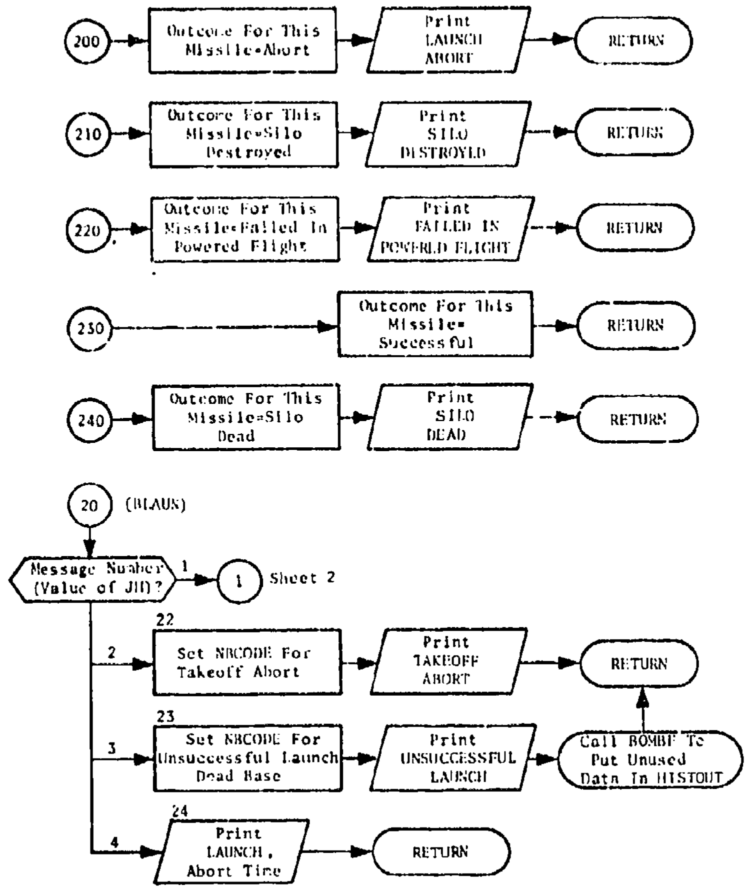


Fig. 25. (cont.)
(Sheet 4 of 13)

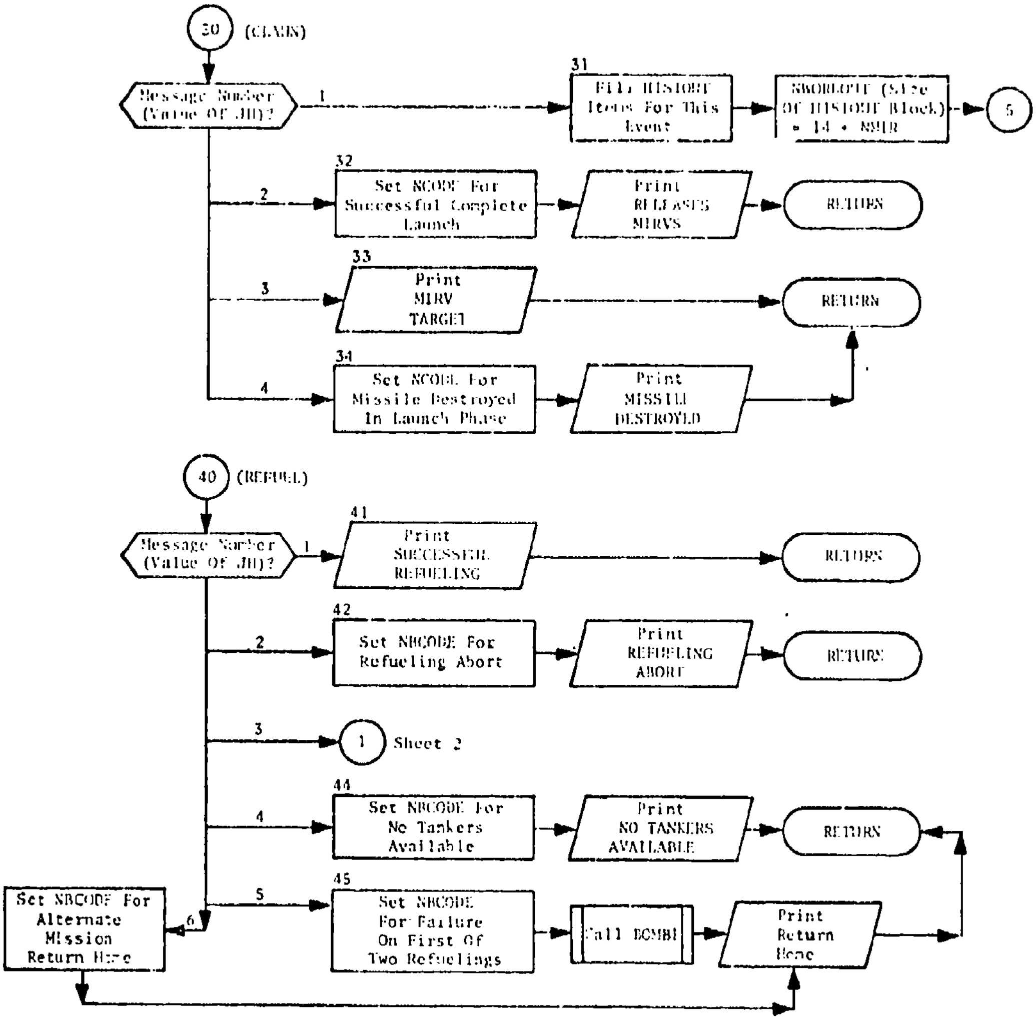


Fig. 25. (cont.)
(Sheet 5 of 13)

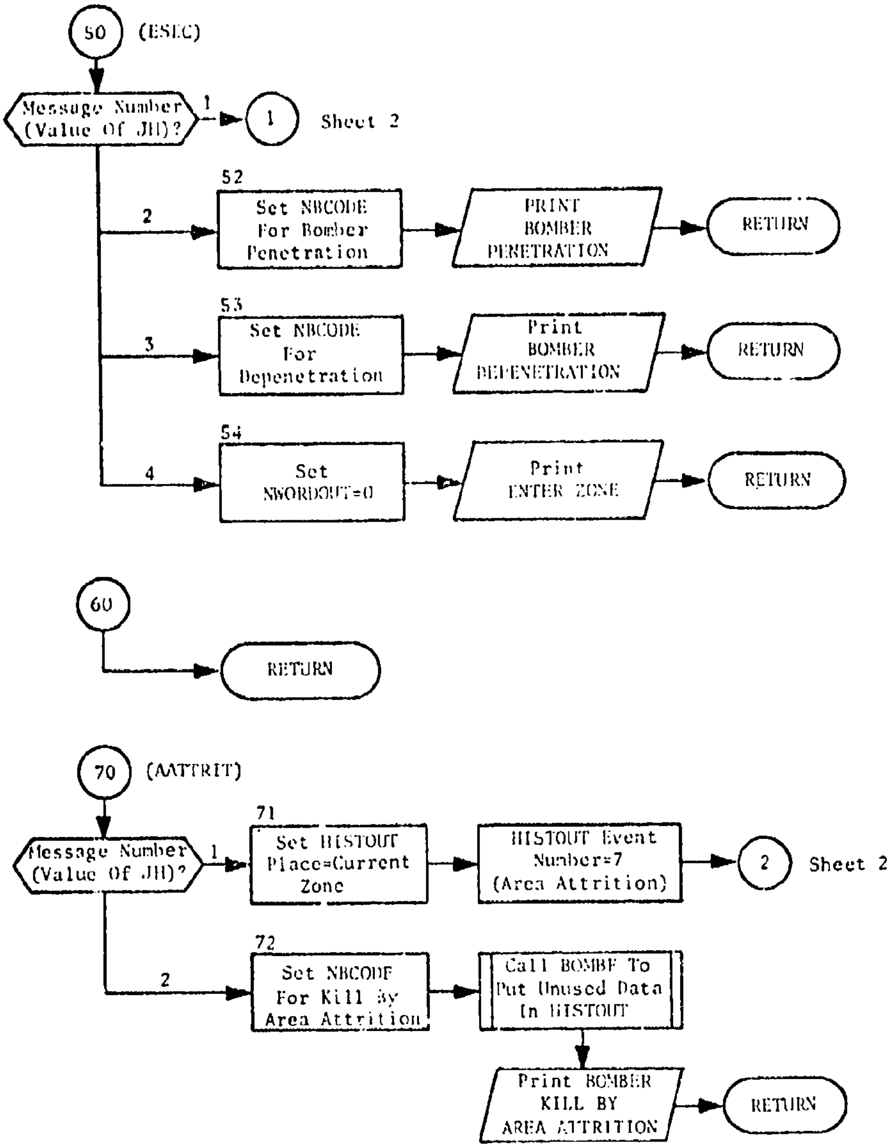


Fig. 25. (cont.)
(Sheet 6 of 13)

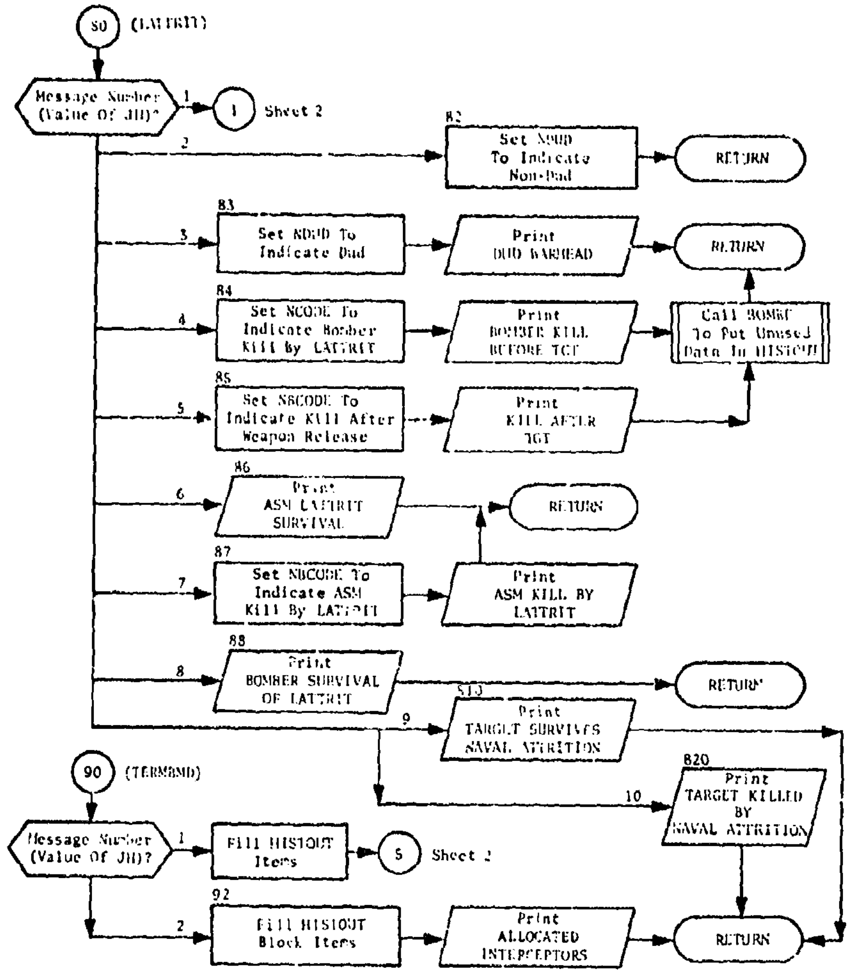


Fig. 25. (cont.)
(Sheet 7 of 13)

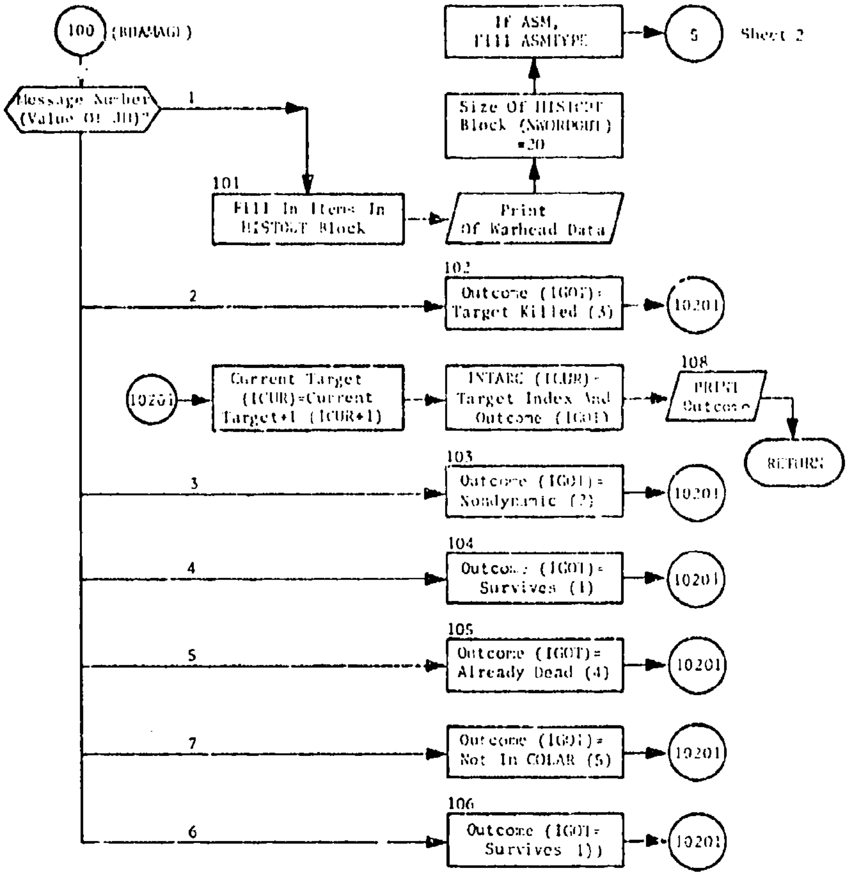


Fig. 25. (cont.)
(Sheet 8 of 13)

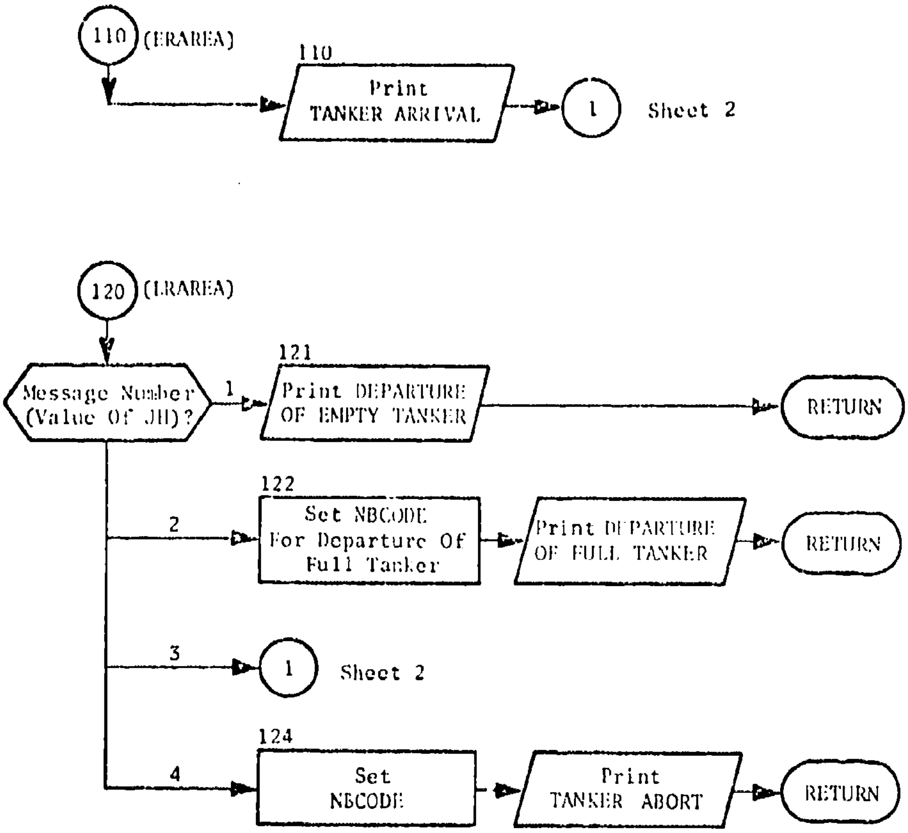


Fig. 25. (cont.)
(Sheet 9 of 13)

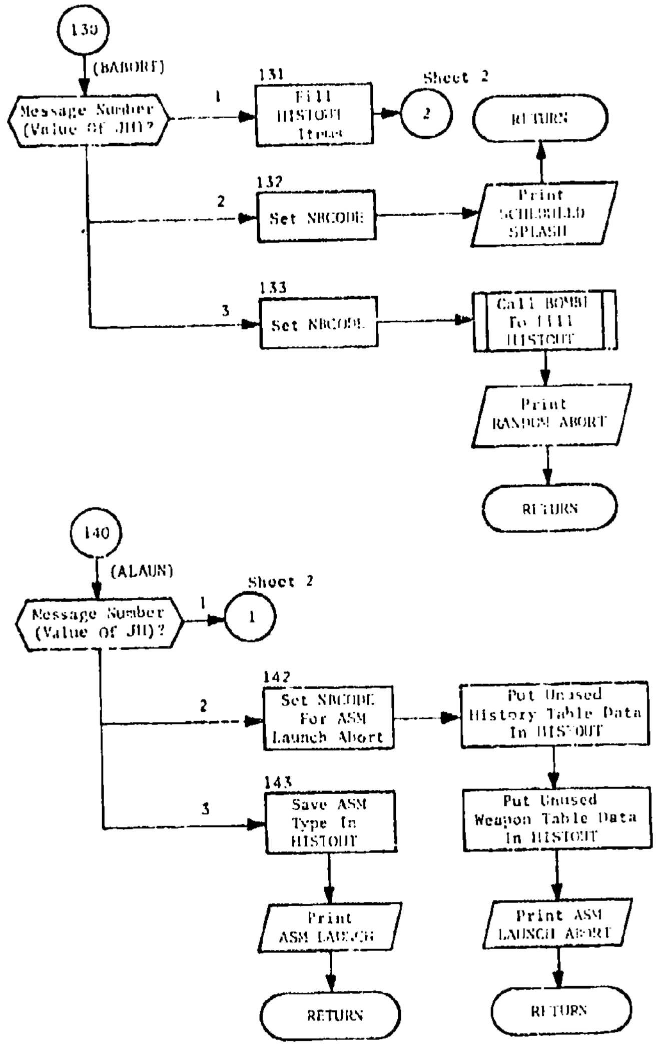


Fig. 25. (cont.)
(Sheet 10 of 13)

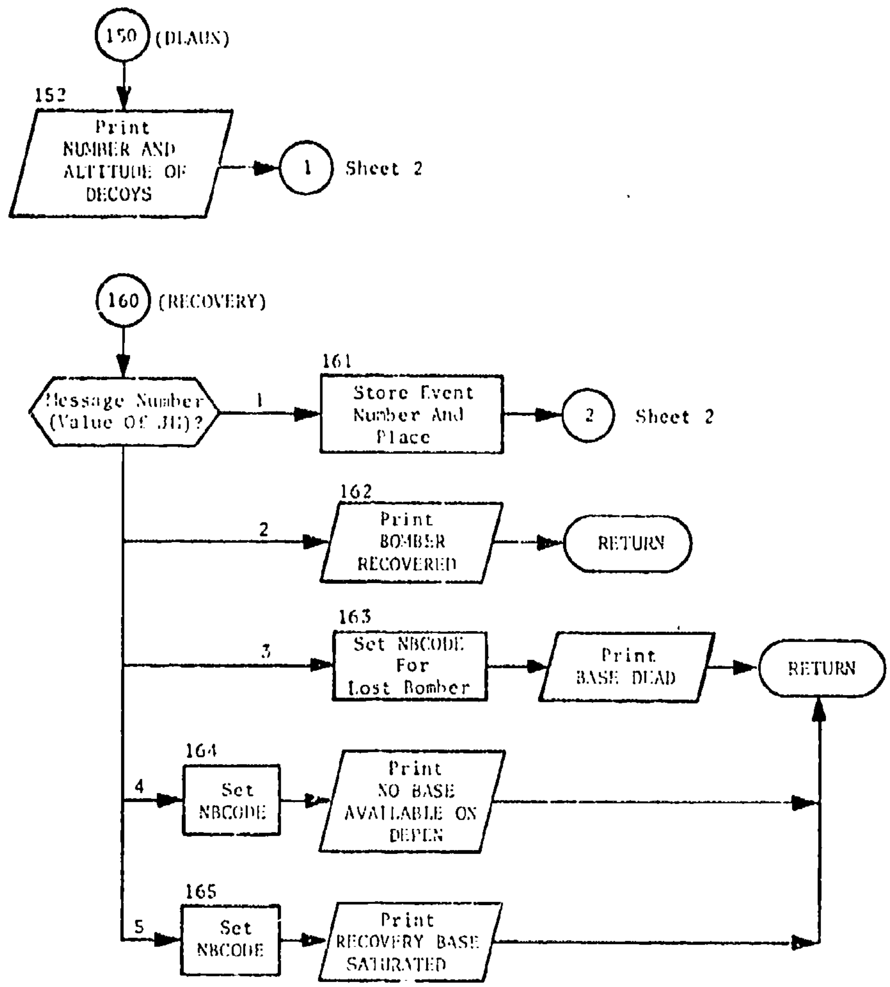


Fig. 25. (cont.)
(Sheet 11 of 13)

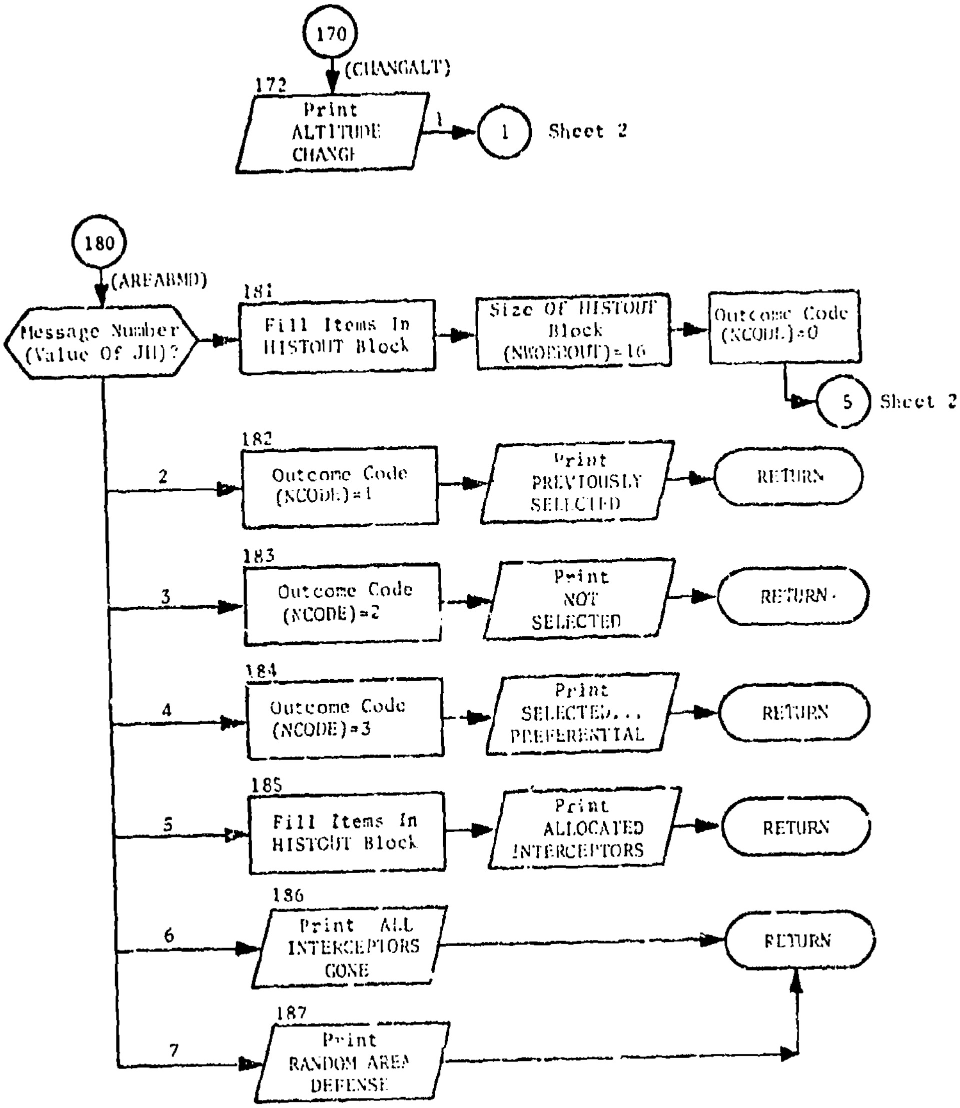


Fig. 25. (cont.)
(Sheet 12 of 13)

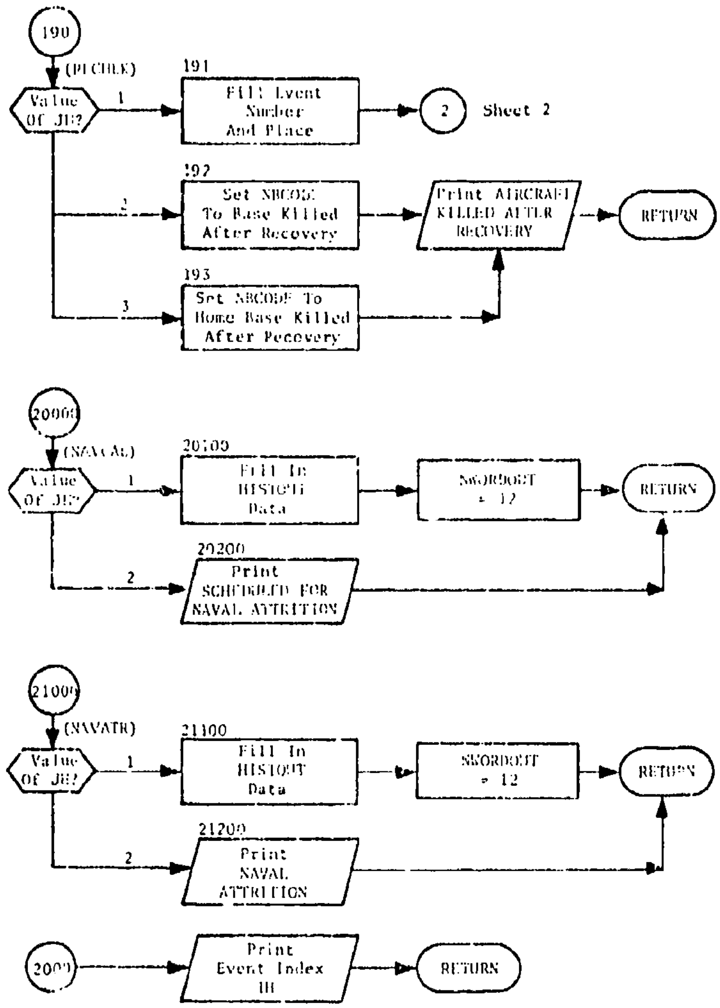


Fig. 25. (cont.)
(Sheet 13 of 13)

SUBROUTINE HISTWRIT

PURPOSE:	To write out the /HISTOUT/ block onto the History tape.
ENTRY POINTS:	HISTWRIT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	HISTOUT, ITP, NWORDOUT, TWORD, TIME
SUBROUTINES CALLED:	WRARRAY, WRWORD
CALLED BY:	AREABMD, BDAMAGE, DONEXT, LRAREA

Method

ITP, the current tape unit, is set to indicate the History tape. ITWORD is set to NWORDOUT, the number of words of the /HISTOUT/ block to be written out. Subroutine WRWORD is called to write ITWORD. If the vehicle is a bomber or tanker, FUTIME is written in HISTOUT(4). Subroutine WRARRAY is called to write NWORDOUT words of the /HISTOUT/ block.

The first NWORDOUT words of the /HISTOUT/ block are then set to zero, and the subroutine exits.

Subroutine HISTWRIT is illustrated in figure 26.

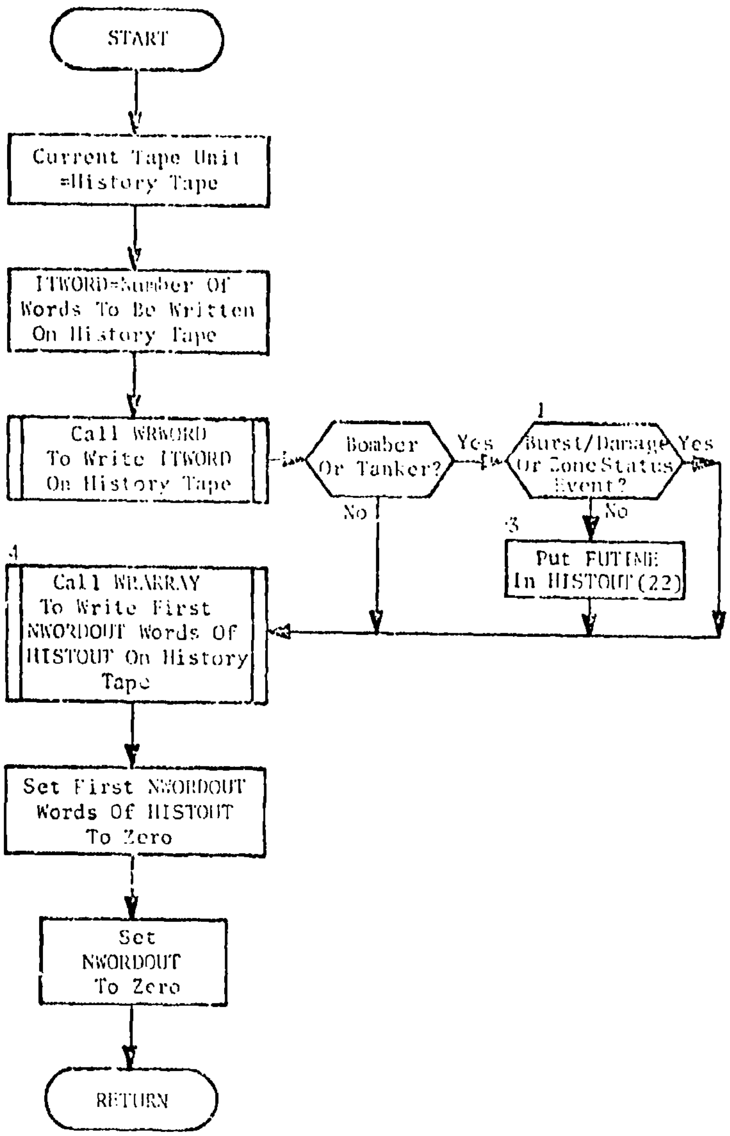


Fig. 26. Subroutine HISTWRIT

FUNCTION IFIND

PURPOSE:	To search array COLAR to find the location of the current target.
ENTRY POINTS:	IFIND
FORMAL PARAMETERS:	IX, the target index
COMMON BLOCKS:	NCOL, 19501
SUBROUTINES CALLED:	None
CALLED BY:	AREABMD, BDAMAGE, NAVCAL

Method

The target index is compared with the leftmost 14 bits of each element in the array COLAR. If a match is found, the corresponding index to the array is returned. If not, a zero is returned.

Function IFIND is illustrated in figure 27.

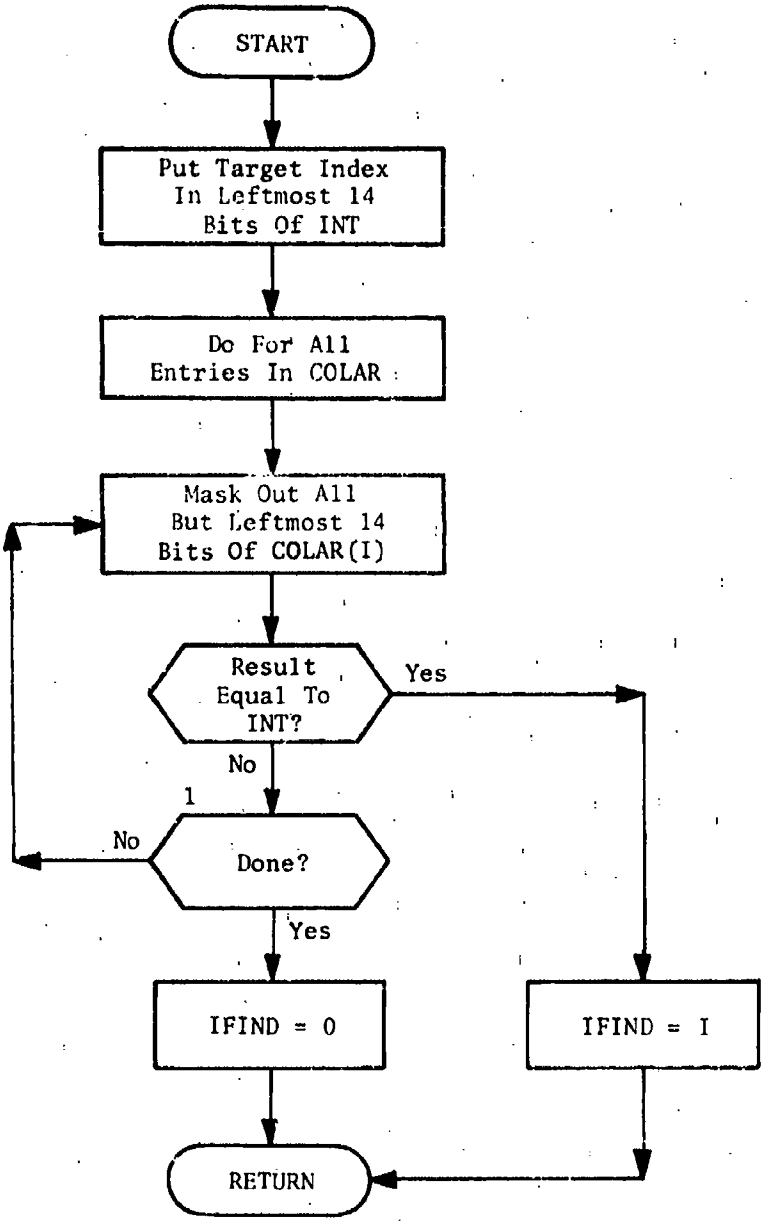


Fig. 27. Function IFIND

SUBROUTINE INITEV

PURPOSE: To initialize elements in common blocks /CONST/, /ESTOR/, and /TIME/.

ENTRY POINTS: INITEV

FORMAL PARAMETERS: ITAPES - Array which specifies the tapes available for spilling

COMMON BLOCKS: CONST, ESTOR, TIME

SUBROUTINES CALLED: None

CALLED BY: SIMULATE

Method

Various constants are set. TAPET(I), the earliest time on a spill tape, is set equal to a large number for all possible tapes. ITOGO(I), the number of words on a spill tape, is set equal to zero for all active tapes and -1 for inactive tapes.

Subroutine INITEV is illustrated in figure 28.

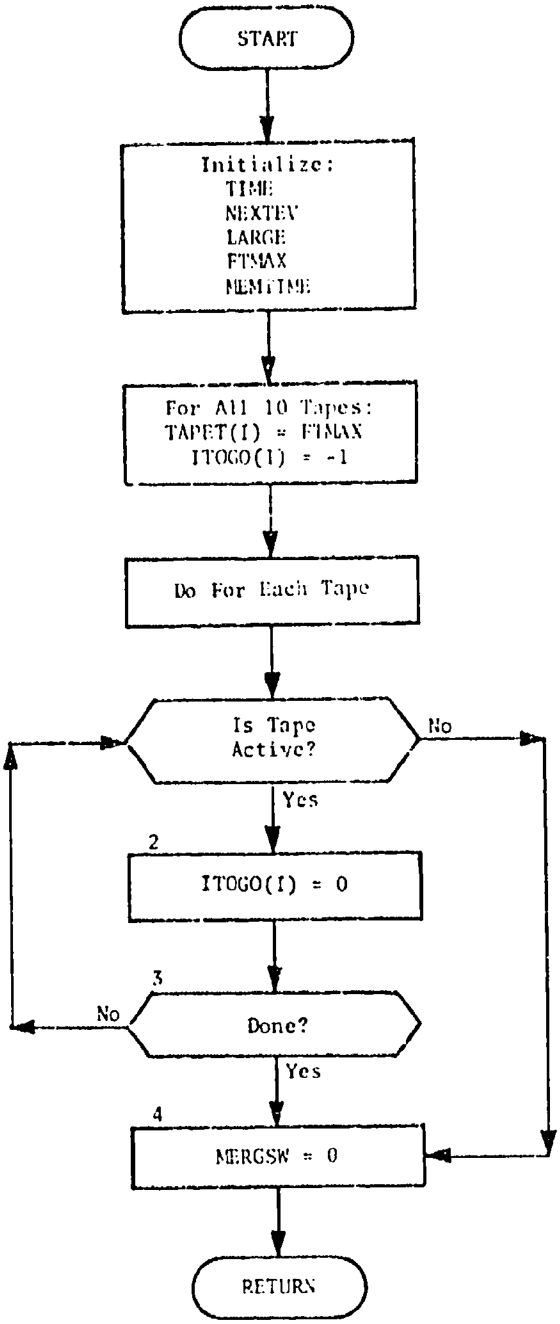


Fig. 28. Subroutine INITEV

SUBROUTINE INITLIST

<u>PURPOSE:</u> To initialize list memory.

<u>ENTRY POINTS:</u> INITLIST

<u>FORMAL PARAMETERS:</u> None

<u>COMMON BLOCKS:</u> ESTOR, EVINDX, LISTMEM, MONDAT, 19501

<u>SUBROUTINES CALLED:</u> ADDMEM

<u>CALLED BY:</u> SIMULATE

Method

MONSW and IAVAIL are set equal to zero. INITLIST initializes list memory by adding the first 3,000 words of the array IARR to list memory by a call on ADDMEM. In addition, it initializes the arrays EVTIME and INDEVB.

Subroutine INITLIST is illustrated in figure 29.

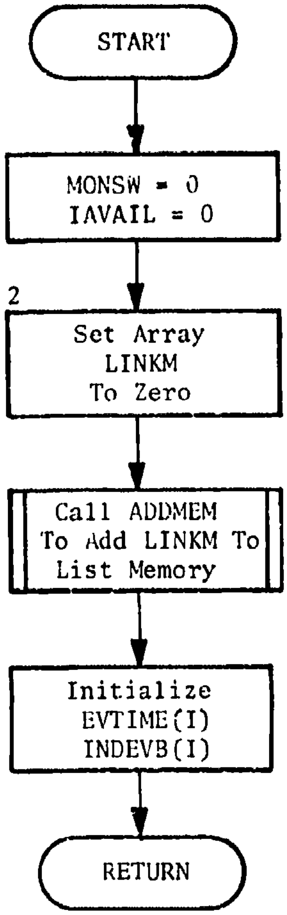


Fig. 29. Subroutine INITLIST

SUBROUTINE LATTRIT

PURPOSE:	To simulate a bomber penetrating local defenses.
ENTRY POINTS:	LATTRIT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ASMS, BOMBER, EDATA, GROUND, KEYWORDS, LATTRIT, TIME, WARHEAD, ZONES, 19501
SUBROUTINES CALLED:	BDAMAGE, EXPF,* HIST, IGET, NEXTEVNT, PLANTS, RANF,* RNDEV
CALLED BY:	DONEXT

Method

The execution of LATTRIT is recorded through a call to subroutine HIST. IGO, which indicates if there is to be a follow-on event, is set to zero. The target index INDP(IHT) is retrieved from the History Table and stored in INTAR.

If the weapon is a naval attack weapon, a separate test is made to determine whether the weapon penetrates and, if it does, to determine PKNAV, the penetration probability being stored in INDATA. For non-naval attack delivery vehicles, IALT, the vehicle altitude index, is tested. If the vehicle is high, the target's number of high-altitude defense batteries is retrieved, multiplied by DEFHI the effectiveness factor for high-altitude defenders, and stored in DEF. If the vehicle is low, TARDEFLO, the index to the target's number of low-altitude defense batteries, is retrieved, multiplied by DEFLO the effectiveness factor for low-altitude defenders, and stored in DEF. The number of accompanying decoys at the relevant altitude is stored in NDECOYS.

If DEF is nonzero, the target is defended. The survival probability SP is computed as the exponential function of -DEF/(1+.5*NDECOYS). SP is compared to a random number generated by library subroutine RANF to determine if the vehicle survives.

If the vehicle survives or if DEF is zero, the type of vehicle undergoing attrition is found by testing the class indicator. If ICLASS is greater than two, the vehicle is an air-to-surface missile (ASM). Survival of the ASM is recorded.

---

*System Library Function

If ICLASS equals two, the vehicle is a bomber. All low-altitude decoys are considered downed, so that the number of penetrators in the current zone NPENZ(INZONE) is decreased by NDLO, which is then set to zero. Since the bomber survived, IGO is set equal to one.

Whether the vehicle is a bomber or an ASM, the weapon type IWTYP(JWT) is retrieved from the Weapon Table and stored in NWTYP. The dud probability of this type PDUD(NWTYP) is compared to a random number to determine if the weapon is a dud. If so, this is recorded, and the subroutine exits.

If the weapon detonates, the class indicator ICLASS is tested. If ICLASS is greater than two, the vehicle is an ASM, and SIG is computed by multiplying the CEP for this type of ASM by a constant. If ICLASS is two, the vehicle is a bomber, and SIG is computed by multiplying the CEP for this type of bomber by a constant. The components of actual ground zero, IAGX and IAGY, are computed by multiplying SIG by the value returned by function RNDEV and adding the desired ground zero components. Non-dud weapon release is recorded, and subroutine BDAMAGE is called to assess the results. If IGO is one (live bomber), subroutine NEXTEVNT is called; otherwise, a return is made directly.

The release of a live weapon is recorded, and a call to BDAMAGE is made. The Weapon Table pointer is then incremented by one. If the weapon was delivered by a surviving bomber, subroutine NEXTEVNT is called to plant the next event for that bomber.

If the vehicle does not survive, the class indicator is tested to determine the vehicle type. If ICLASS equals three, the vehicle is an ASM. ASM kill by local attrition is recorded. If ICLASS equals two, the vehicle is a bomber. The number of penetrators in the current zone is decreased by one for the bomber plus the number of active decoys. The bomber kill is recorded. The probability of bomber kill before weapon release BKBRP is compared to a random number to determine if the kill bomber successfully released a weapon. If so, this is recorded and control passed to the point above at which the weapon desired-ground-zero components are retrieved from the Weapon Table.

Subroutine LATTRIT is illustrated in figure 30.

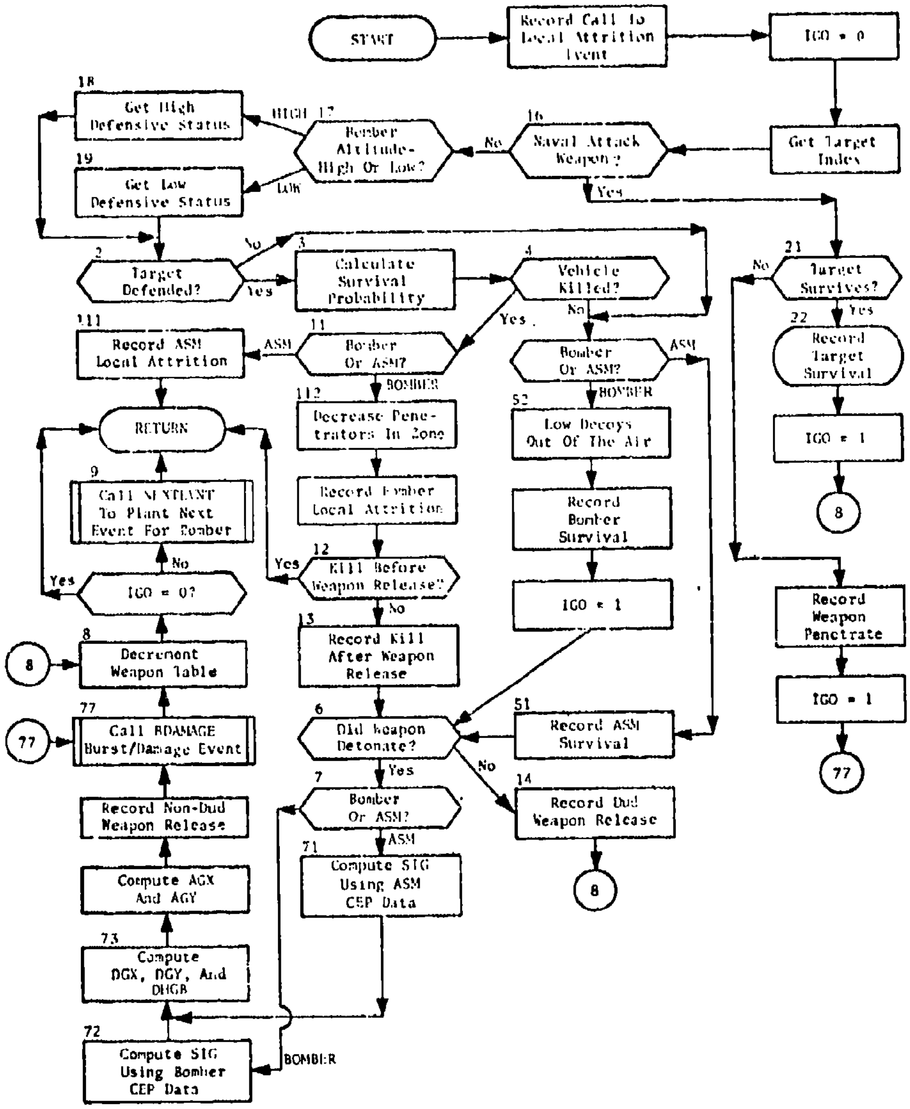


Fig. 30. Subroutine LATTRIT

SUBROUTINE LRAREA

PURPOSE:	To simulate the departure of a tanker from a refueling area.
ENTRY POINTS:	LRAREA
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, EPSN, HISTOUT, HISTREF, KEYWORDS, RECOV, REFUEL, TIME, 19501
SUBROUTINES CALLED:	HIST, HISTWRIT, PLANTS, RANF*
CALLED BY:	DONEXT

Method

The execution of LRAREA is recorded through a call to subroutine HIST. The refueling area index INDP(IHT) is retrieved from the tanker History Table and stored in INDRA.

If TIME, current game time, equals TABORT, tanker in-flight abort time, a tanker abort has occurred. Then the fraction of total tankers in the area that are empty is compared to a random number generated by subroutine RANF to determine if an empty tanker will leave the area. If not, the amount of fuel is subtracted from the amount in the area, the abort is recorded, and the subroutine exits.

If there has been no abort, the number of empty tankers in this area NETANK(INDRA) is tested. If there are empty tankers here or if an empty tanker had aborted, the array is decreased by 60 and the departure of an empty tanker is recorded. If the departure is due to abort, this fact is recorded and the subroutine exits.

If there are no empty tankers in the area, the amount of fuel in the area NFTANK(INDRA) is decreased, and the departure of a tanker is recorded.

If the departing tanker has not aborted, a recovery base is selected from the list carried in the tanker's History Table, and a Recovery event is planted unless an enroute abort occurs first.

Subroutine LRAREA is illustrated in figure 31.

*System Library Function

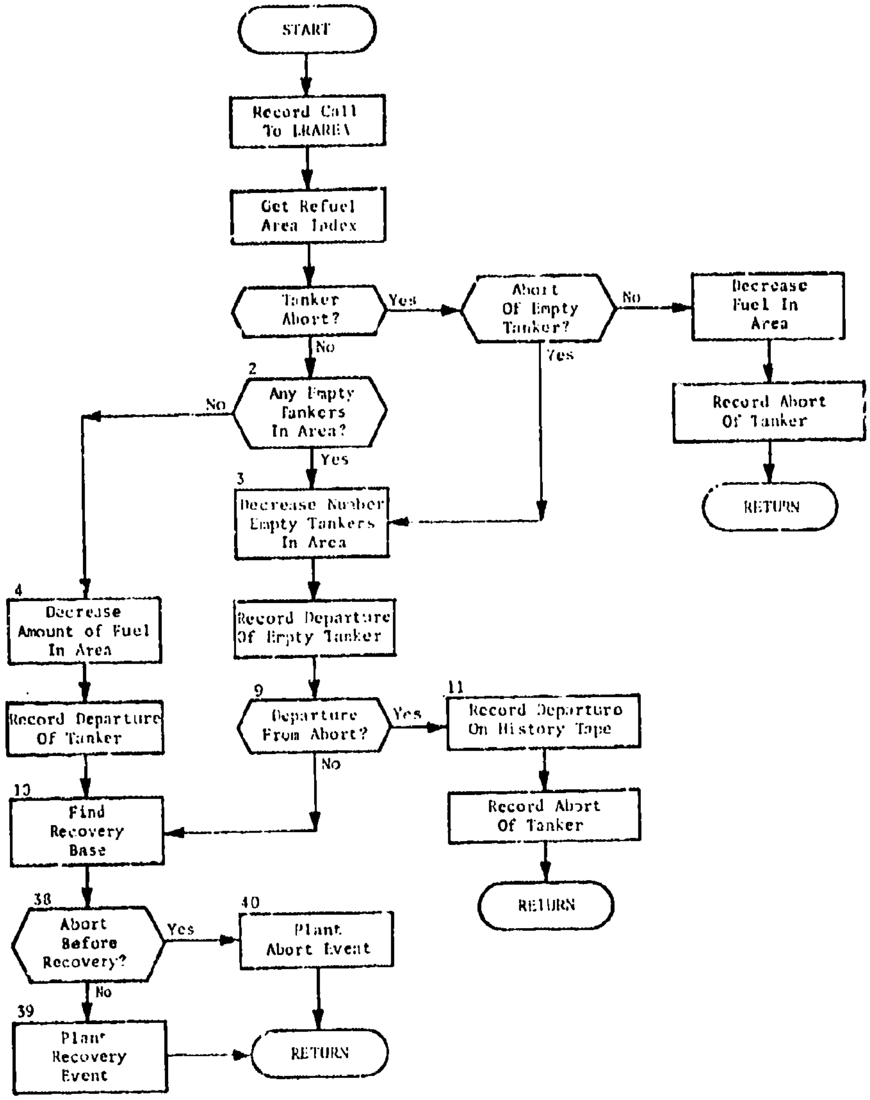


Fig. 31. Subroutine LRAREA

SUBROUTINE MLAUN

PURPOSE:	To simulate the launch of one or more missiles from a single squadron.
ENTRY POINTS:	MLAUN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, HISTA1, INDEX, INREP, KEYWORDS, MISSLE, TIME, TRYL, 19501
SUBROUTINES CALLED:	HIST, IGET, IPUT, PLANTS, RANORDER, TRYLAUN
CALLED BY:	DONEXT

Method

The missile type data for this event are stored in items of common block /TRYL/ for the call to TRYLAUN. The execution of subroutine MLAUN is recorded through a call to subroutine HIST. The variable IREP is set equal to one unless the reprogramming option has been called for. The missile launch result counters are initialized to zero and subroutine TRYLAUN is called to try to put the missile into powered flight.

If after this call the number of missiles processed NCYCLE equals the number of missiles in the event NWPNS, this is recorded. If missiles remain, the number of targets processed NTEST is tested to see if NTARG the number of targets allotted to this squadron have been covered. If any targets remain, TRYLAUN is again called. If none remain, this is recorded.

When either all missiles have been processed or all targets covered, NSUCC is tested to see if there are any successful launches. If so, subroutine RANORDER is called to randomly reorder the ordered target list. The targets are matched with missiles, weapon and launch base data moved from INDATA to OUTDATA, the successful missile launches recorded, and Complete Launch events planted.

Whether or not there are any successful launches, NLO, the number of missiles left over, is tested to see if all missiles are processed. If so, the event is finished. If not, NLATER is tested to see if any unsuccessful missiles can be launched later. If none can be, the event is finished. Otherwise, INDATA elements are moved to OUTDATA, new Launch events planted for a time equal to game time TIME plus the time required to retarget TRETARG, and the planting to the new events recorded. The subroutine then exits.

Subroutine MLAUN is illustrated in figure 32.

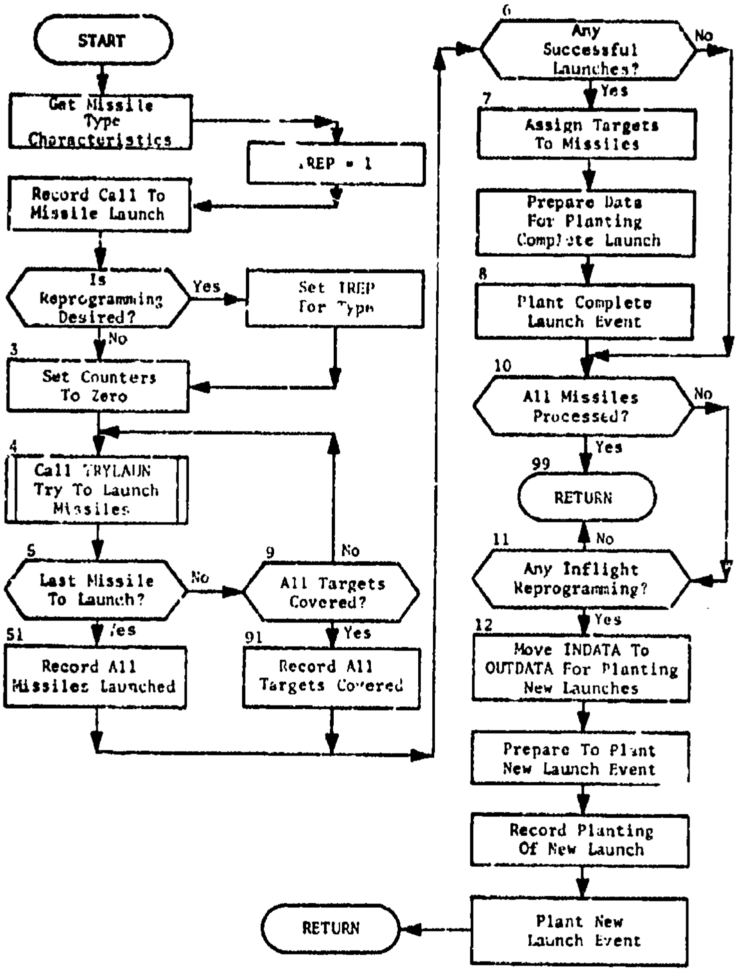


Fig. 32. Subroutine MLAUN

SUBROUTINE MONPRIN

PURPOSE:	To initialize and control the printing of event data at execution time.
ENTRY POINTS:	MONPRIN, OPTPRIN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, EVENT, MONDAT, NSW, TIME
SUBROUTINES CALLED:	PLANT
CALLED BY:	RDCARDS, DONEXT

Method

Entry OPTPRIN is called to initiate the diagnostic print sequence. The NSW cards containing values for X and Y, one pair to a data card, are read and stored. X is the beginning time and Y is the ending time of a period in which diagnostic printing is desired. MHT, the number of lines in the History Table, is set to 2*NSW, since each of the NSW time cards contains two values. IHT, the current History Table line, is set to one.

MONSW, the print switch tested by subroutine DONEXT, is turned off (+1). The time limits X(I) and Y(I) are stored alternately in History Table array items TINC(2*I-1) and TINC(2*I), I=1, NSW. Control is transferred to the point below at which a new MONPRIN execution is planted.

Entry MONPRIN is called after the subroutine has been initialized by OPTPRIN. MONSW is turned on if it is off, or off if it is on. Since the X(I) and Y(I) are stored alternately, the result is that MONSW is turned on at the times X(I) and turned off at the times Y(I). A value of +1 is off; -1 is on. If IHT equals 1, no further executions of MONPRIN are desired, so the subroutine exits. Otherwise, IHT is moved to point to the next time in the History Table.

A new MONPRIN execution is planted. FUTIME, event execution time, is set to the current time in the History Table. INDATA is stored in OUTDATA, subroutine PLANT is called to plant an execution of MONPRIN which uses the bomber INDATA structure, and the subroutine exits.

Subroutine MONPRIN is illustrated in figure 37.

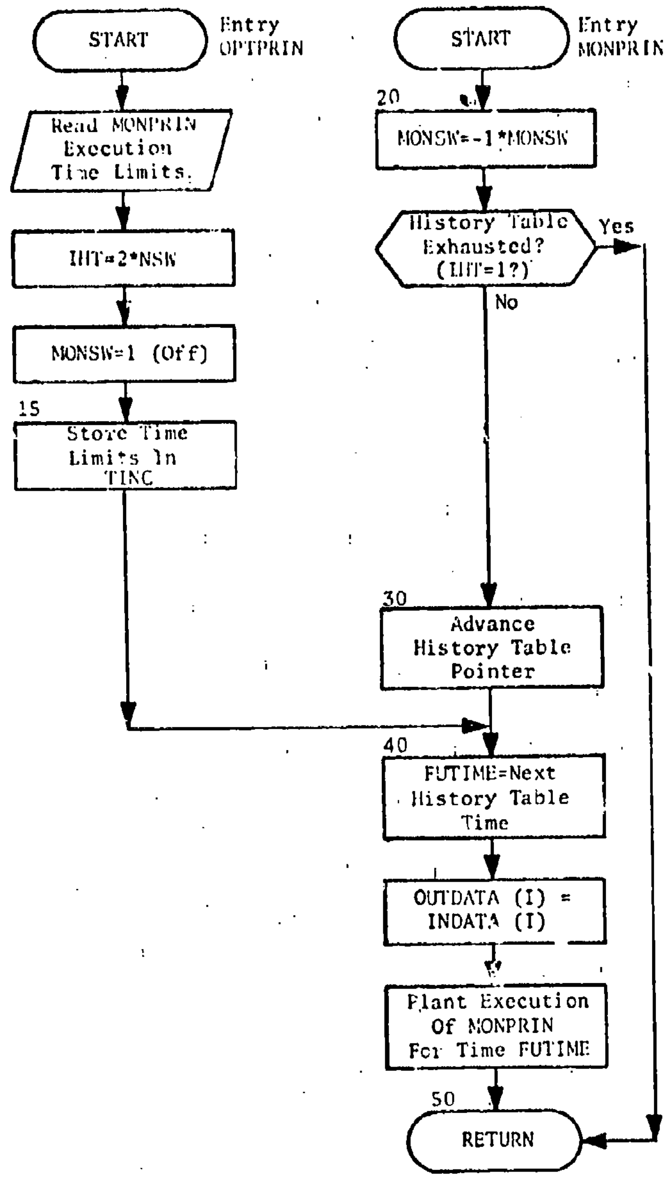


Fig. 33. Subroutine MONPRIN

SUBROUTINE MSINPRIN

PURPOSE: To print out INDATA for a missile launch event.

ENTRY POINTS: MSINPRIN

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA, IPSWICH

SUBROUTINES CALLED: MUNPK

CALLED BY: SIMULATE

Method

This subroutine is called from SIMULATE only if IPSW1 is set equal to one through data read in on cards by subroutine RDCARDS. The first 14 words of INDATA are printed, followed by the identifiers of the missiles to be launched. If IPSW4 is equal to one, the target list and times of flight are printed out. If IPSW5 is equal to one, the target aim coordinates and number of terminal and area decoys for each target are printed out. The target information is unpacked from INDATA through a call to subroutine MUNPK.

Subroutine MSINPRIN is illustrated in figure 34.

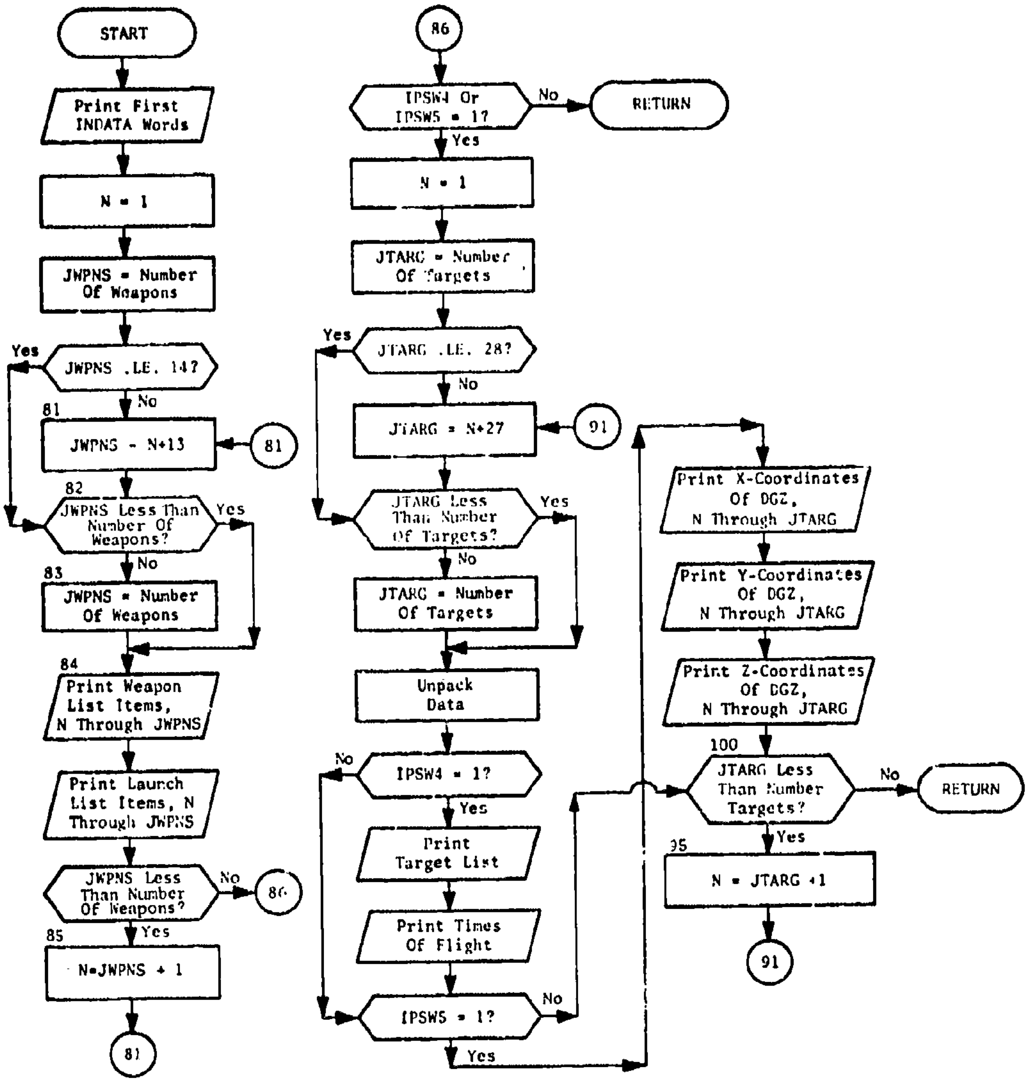


Fig. 34. Subroutine MSINPRIN

SUBROUTINE MUNPK

PURPOSE: To unpack missile data from INDATA.

ENTRY POINTS: MUNPK

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA

SUBROUTINES CAL''D: None

CALLED BY: CLAUN, MSINPRIN

Method

The target data for missiles include a word of packed information for each target. The word has the following structure:

Bits	15	12	12	3	3	3
Variable	Target Index	DGX	DGY	DHOB	Area Decoys	Term Decoys

By successive masking and right-shifting, the variables are separated, from right to left, and stored in the appropriate separate locations of INDATA or OUTDATA. If DGX or DGY is negative, the leftmost bit is extended. The total number of area and terminal aim points is determined by adding in the number of warheads.

Subroutine MUNPK is illustrated in figure 35.

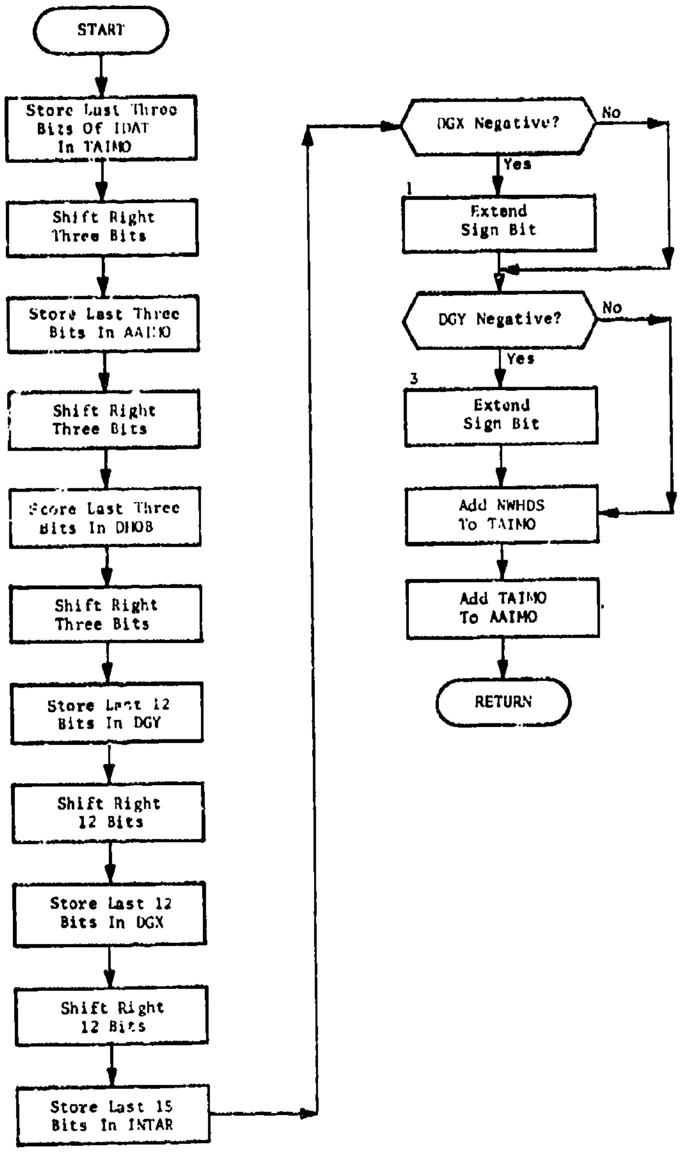


Fig. 35. Subroutine MUNPK

SUBROUTINE NAVATR

PURPOSE: To kill a target by naval attrition.

ENTRY POINTS: NAVATR

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA, KEYWORDS, 19501

SUBROUTINES CALLED: HIST, IPUT

CALLEY BY: DONEXT

Method

The attrition is recorded by two calls to subroutine HIST. The target, and those collocated with it, ar killed with calls to subroutine IPUT.

Subroutine NAVATR is illustrated in figure 36.

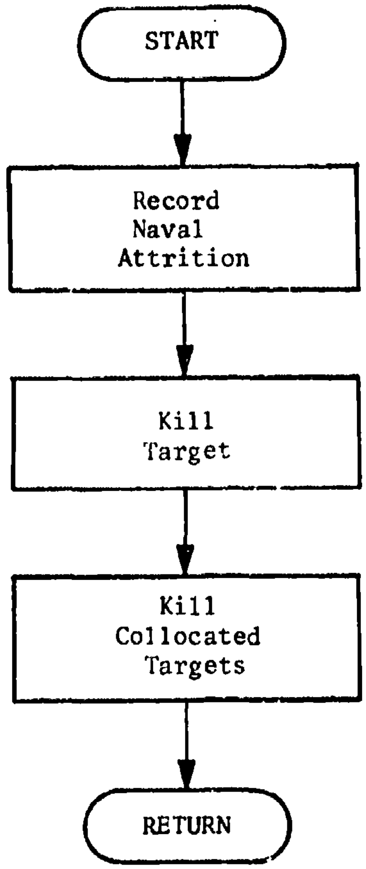


Fig. 36. Subroutine NAVATR

SUBROUTINE NAVCAL

PURPOSE:	To determine time of naval attrition.
ENTRY POINTS:	NAVCAL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, KEYWORDS, NAVDATA, NWORDOUT, TIME, 19501
SUBROUTINES CALLED:	HIST, IGET, IPUT, RANF,* PLANTS, IFIND
CALLED BY:	DONEXT

Method

The DBL type is taken from INDATA(3) and stored in JD, and a call on HIST is made to record the call to NAVCAL. The DBL status of the base is extracted with IGET and tested to see if a DBL event (NAVATR) for this base has already been planted. If it has, the tape record is suppressed, and the subroutine exits.

If no DBL event has been planted, the DBL status of the base and its collocated parts is set to one. A random number P is selected, and the curve of time-probability of DBL for the current value of JD is analyzed to determine the time at which the probability of DBL reaches P. Linear interpolation between data points is used. If the probability never attains the value P, the History tape record is suppressed, and the subroutine exits. If such a time is found, a NAVATR event is planted to occur at the appropriate time.

Subroutine NAVCAL is illustrated in figure 37.

---

*System Library Function

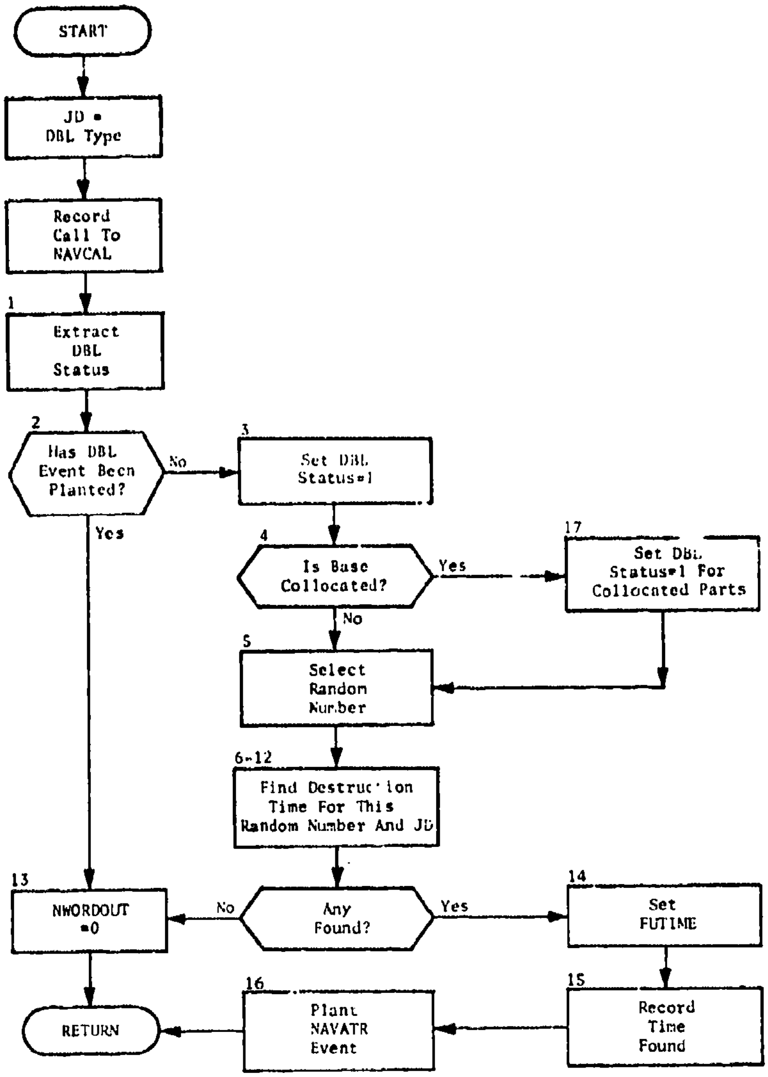


Fig. 37. Subroutine NAVCAL

SUBROUTINE NEXTEVNT

PURPOSE:	If the bomber is in enemy territory, to calculate the probability of attrition by area defenses; and to plant the next event for the vehicle.
ENTRY POINTS:	NEXTEVNT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AATTRIT, DATTA, EDATA, EPSN, TIME, ZONES
SUBROUTINES CALLED:	EXPF,* PLANTS, RANF*
CALLED BY:	ALAUN, BLAUN, CHANGALT, DLAUN, ESEC, LATTRIT, REFUEL

Method

If there are no more events in the History Table, the subroutine returns. Otherwise, the pointer is moved one position. If the vehicle is outside enemy territory or the time to the next event is zero, the next event is from the History Table. If in enemy territory, the probability of surviving enemy defenses is calculated and compared with a random number to determine whether the bomber will suffer attrition on the forthcoming leg. If it is to be killed, the next event is Area Attrition at a random time along the leg. In any case, FUTIME is calculated and compared with the bomber abort time. The appropriate event is then planted, and the subroutine exits.

Subroutine NEXTEVNT is illustrated in figure 38.

---

*System Library Function

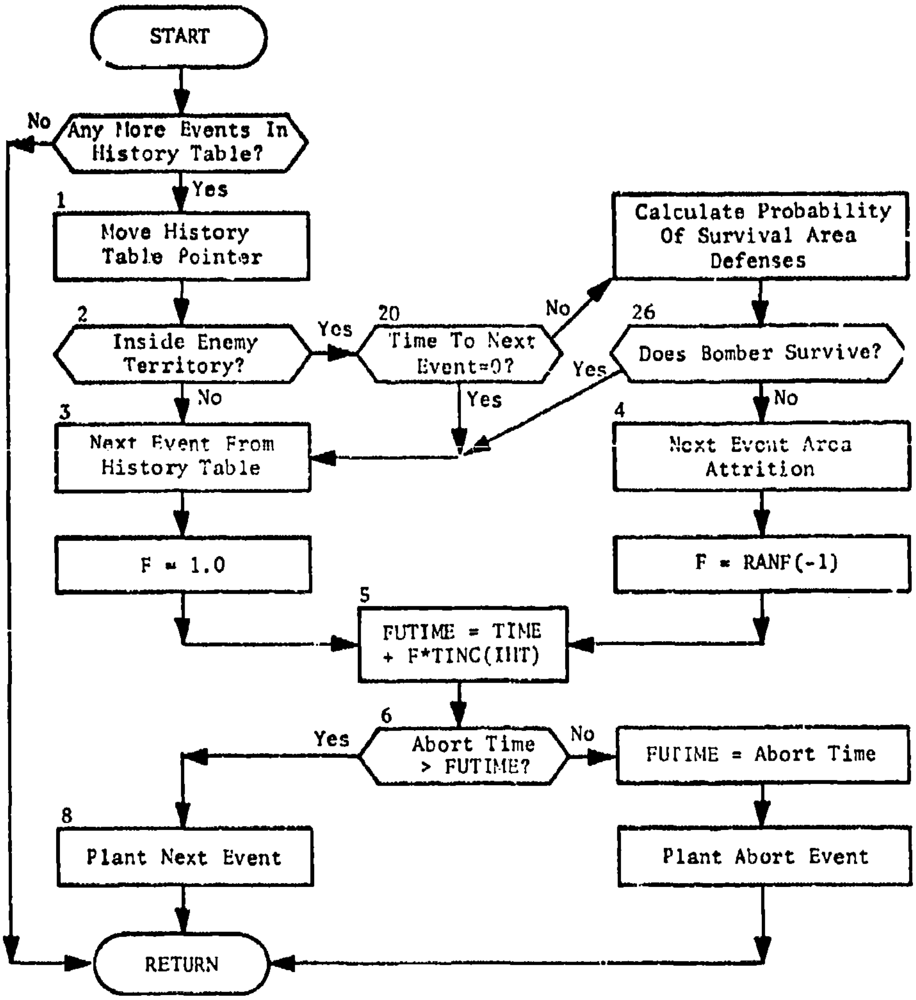


Fig. 38. Subroutine NEXTEVNT

SUBROUTINE PLANT

PURPOSE:	To insert a new event into the internal event store.
ENTRY POINTS:	PLANT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ESTOR, EVENT, EVINDX, LISTMEM, NEVTOT, TIME, 19501
SUBROUTINES CALLED:	ERROL, EVPACK, EVSPILL, UNSQUEEZ, UPDIR
CALLED BY:	MONPRIN, PLANTS, SIMULATE

Method

PLANT begins by calling subroutine EVPACK to pack the event data. If there is no more space in available list memory, subroutine EVSPILL is called to merge the event store with an external file. If the event to be planted is in the past, subroutine ERROL is called to print an error message and terminate the run. If the time of the event to be planted exceeds TMAX, the subroutine returns immediately without planting the event.

If none of the above conditions exists, the header cell of list memory is filled, and subroutine UNSQUEEZ is called to put the event data into available list memory cells. It is then necessary to link the event into the event store. First, the sublist to which the event will be attached is found by comparing the time of the event with the maximum times of the sublists. When the proper sublist is found, the event is inserted, either at the beginning, the end, or the appropriate place in the middle of the sublist. If sublist number one is empty or the sublist entry point directory has not been updated recently, subroutine UPDIR is called, and the subroutine exits.

Subroutine PLANT is illustrated in figure 39.

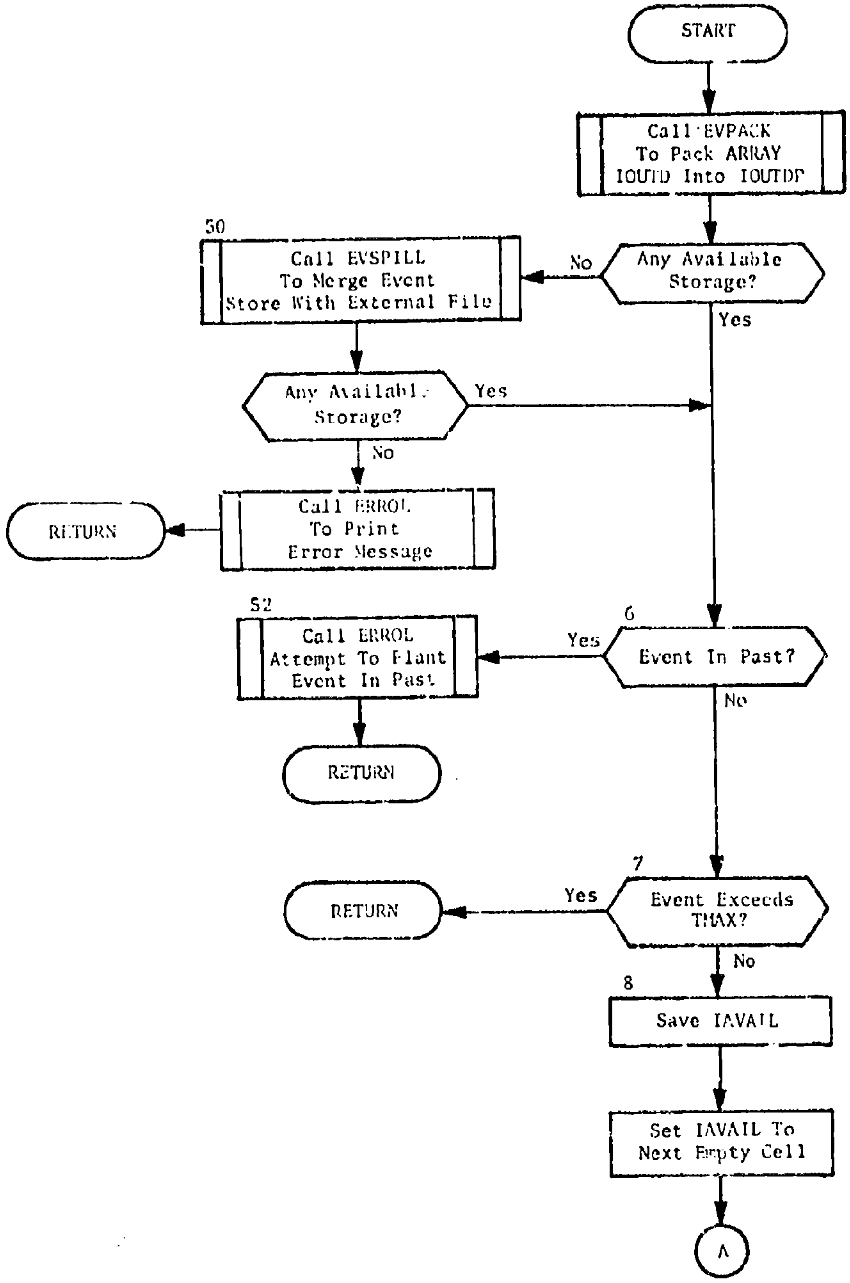


Fig. 39. Subroutine PLANT
(Sheet 1 of 3)

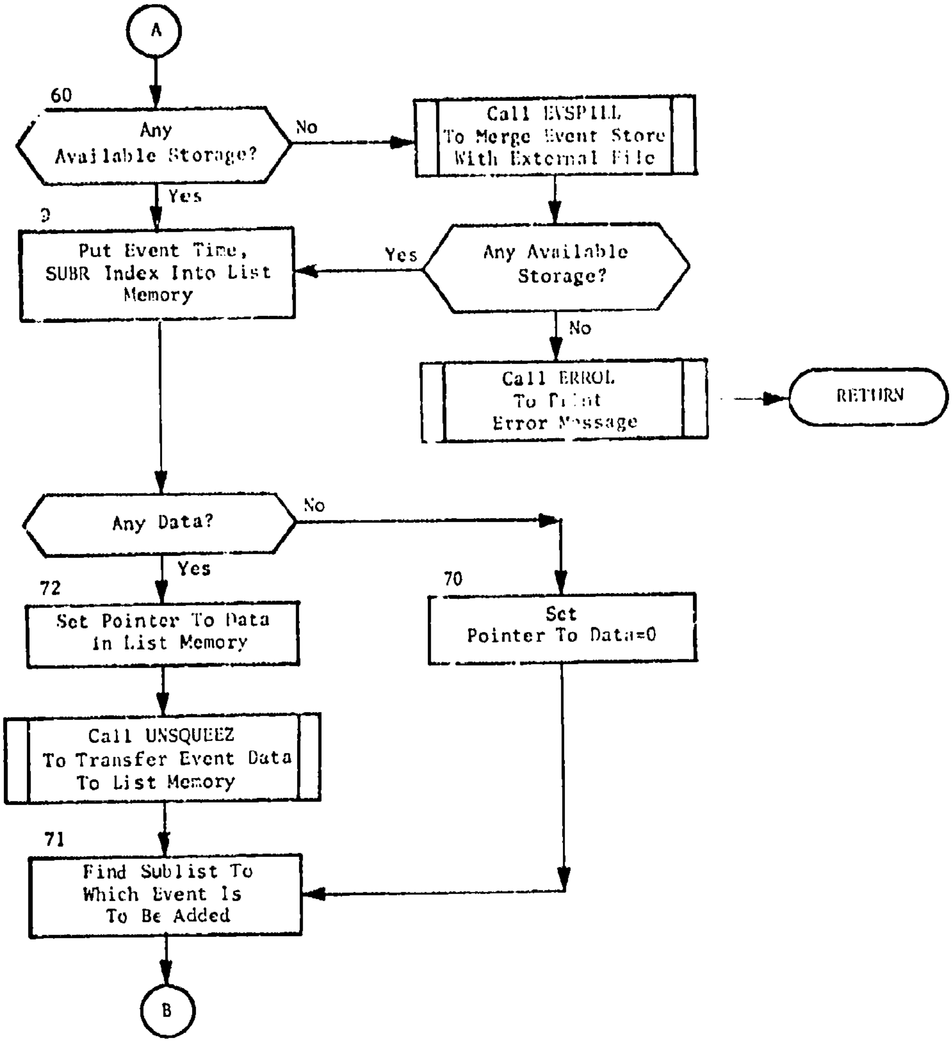


Fig. 39. (cont.)
(Sheet 2 of 3)

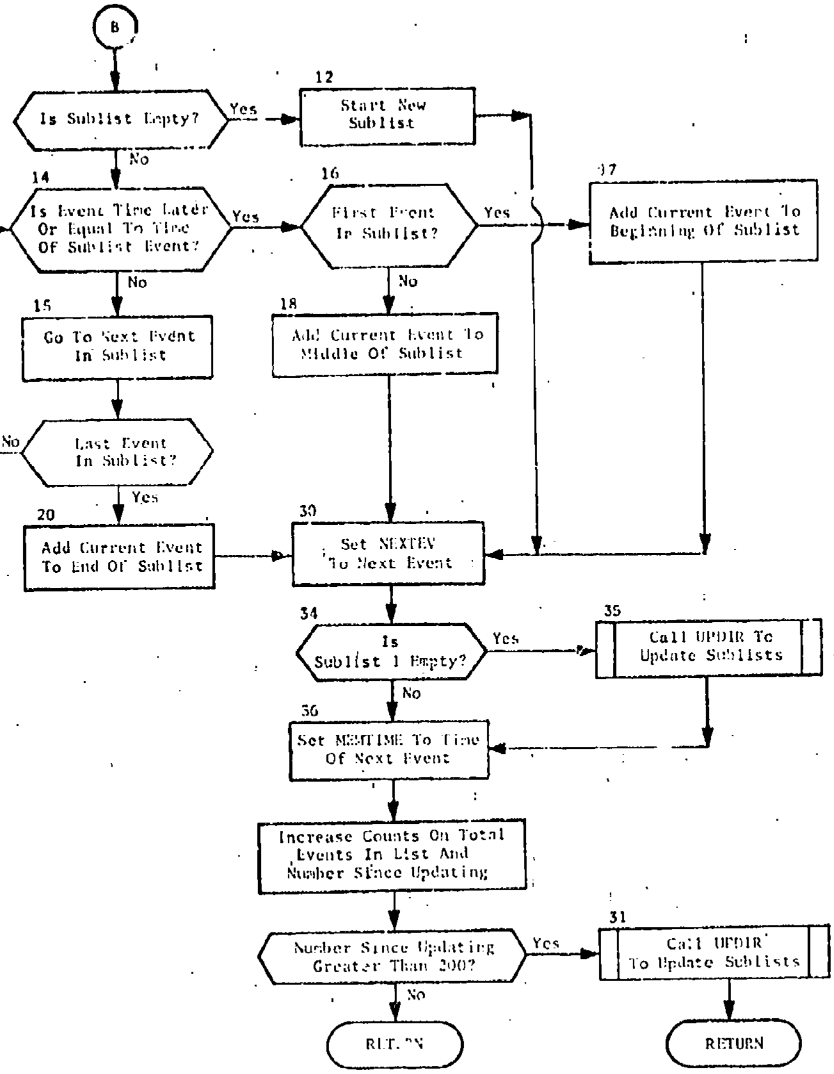


Fig. 39. (cont.)
(Sheet 3 of 3)

SUBROUTINE PLANTS

PURPOSE:	To prepare for planting an event.
ENTRY POINTS:	PLANTS
FORMAL PARAMETERS:	K - Event type to be planted L - Number of words to be moved from INDATA to OUTDATA
COMMON BLOCKS:	DATTA, EDATA, EVENT, FUTPRINT, TIME
SUBROUTINES CALLED:	PLANT
CALLED BY:	ALAUN, BLAUN, CLAUN, ERAREA, ESEC, LATTRIT, LRAREA, MLAUN, NAVCAL, NEXTEVNT, RECOVERY, REFUEL, SIMULATE, SSTAT

Method

The first L words of array INDATA are moved to array OUTDATA. If IFUT is not zero, the execution time of the event being planted is printed. Subroutine PLANT is then called to plant the event.

Subroutine PLANTS is illustrated in figure 40.

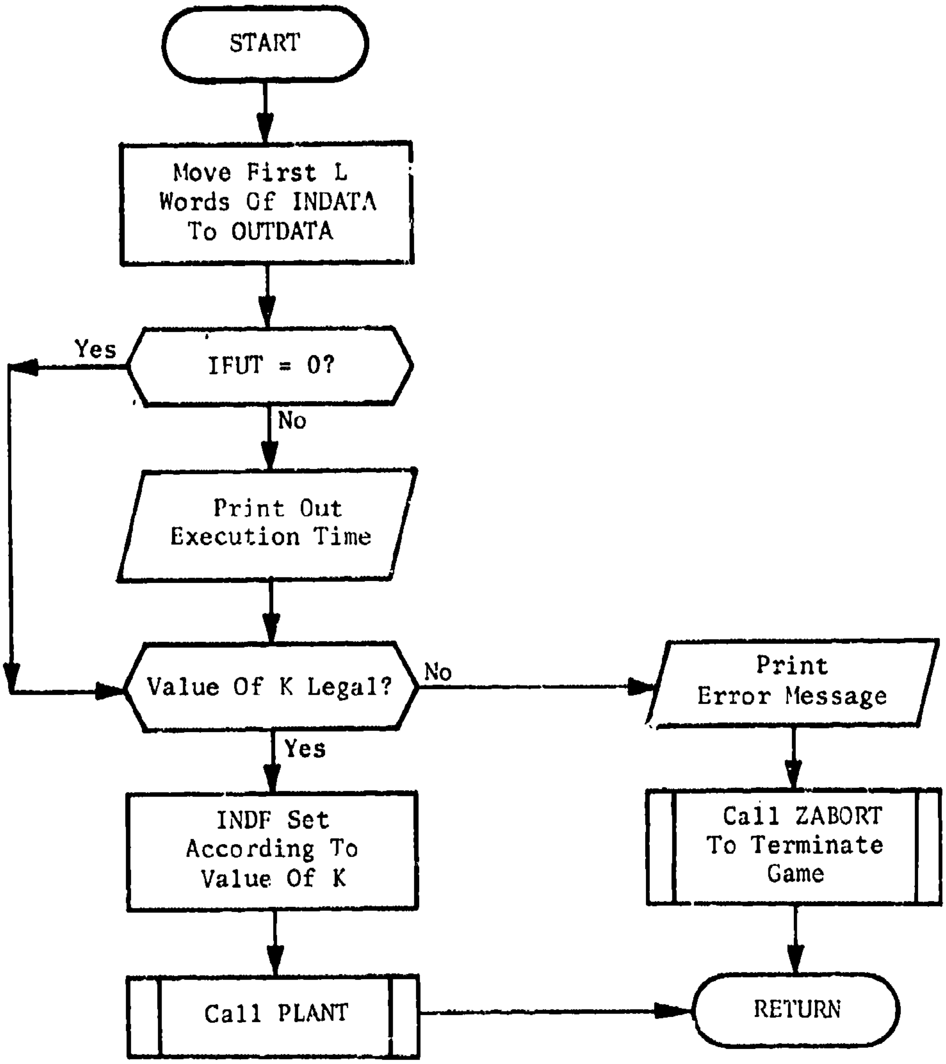


Fig. 40. Subroutine PLANTS

SUBROUTINE PRINDAT

<u>PURPOSE:</u> To print INDATA and INDATAP.

<u>ENTRY POINTS:</u> PRINDAT

<u>FORMAL PARAMETERS:</u> IWDS - Number of words of INDATA to be printed

<u>COMMON BLOCKS:</u> EDATA, ESTOR, TIME

<u>SUBROUTINES CALLED:</u> None

<u>CALLED BY:</u> EVUNPK

Method

TIME, the current game time, is printed. The portion of INDATAP in use is printed, the portion of INDATA in use is printed, and the subroutine exits.

Subroutine PRINDAT is illustrated in figure 41.

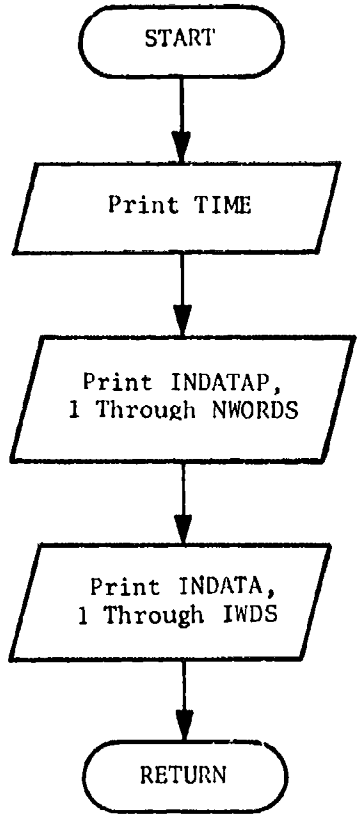


Fig. 41. Subroutine PRINDAT

SUBROUTINE PROUTDAT

PURPOSE:	To print OUTDATA and OUTDATAP.
ENTRY POINTS:	PROUTDAT
FORMAL PARAMETERS:	IUNPKO - Maximum OUTDATA element used
COMMON BLOCKS:	EDATA, ESTOR, TIME
SUBROUTINES CALLED:	None
CALLED BY:	EVPACK

Method

TIME, the current game time, and FUTIME, event execution time, are printed. NN is set to NWORDS+2, where NWORDS is the maximum OUTDATAP element actually used. The portion of OUTDATAP from one to NN is printed, the portion of OUTDATA from one to IUNPKO is printed, and the subroutine exits.

Subroutine PROUTDAT is illustrated in figure 42.

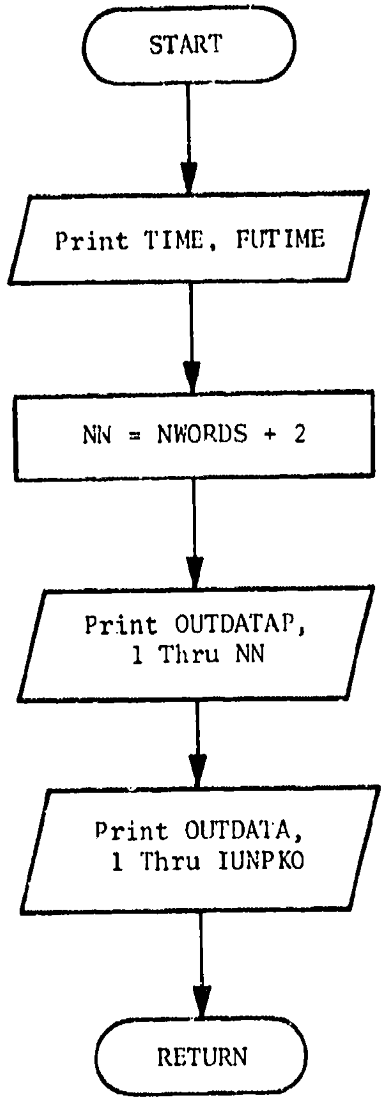


Fig. 42. Subroutine PROUTDAT

SUBROUTINE RANORDER

PURPOSE:	To put the first NTEST integers in random order.
ENTRY POINTS:	RANORDER
FORMAL PARAMETERS:	NTEST - Number of integers to be reordered
COMMON BLOCKS:	INDEX
SUBROUTINES CALLED:	RANF
CALLED BY:	AREABMD, MLAUN, TERMBMD

Method

The first NTEST words of INDEX are assigned integer values, one through NTEST, and the first NTEST words of the array VAL are set to random values. NTEST is the number of integers to be reordered. If no more than one integer is to be reordered, the subroutine returns. Otherwise LI, the number of comparisons to be made on the current pass, is set to (NTEST-1).

If a consecutive pair of random numbers are not in ascending order, the corresponding words of array INDEX are interchanged. On the next pass, one less comparison is required. If after any pass no more comparisons are required, the subroutine exits.

Subroutine RANORDER is illustrated in figure 43.

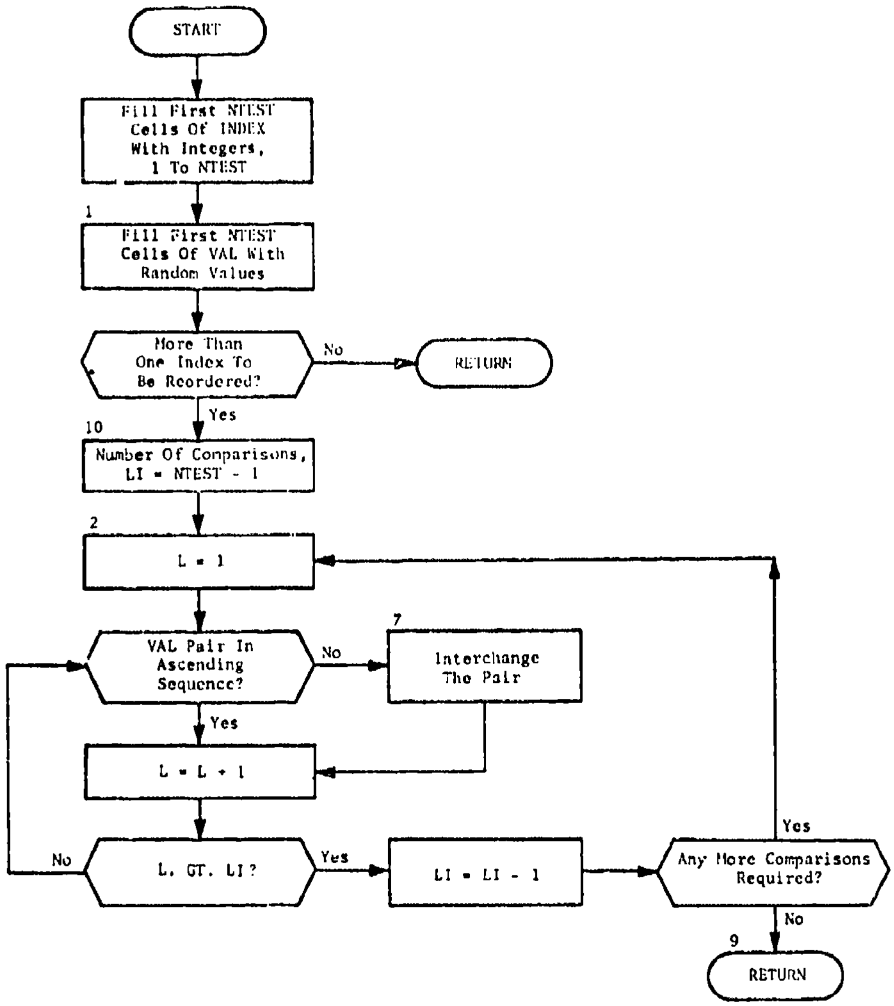


Fig. 43. Subroutine RANORDER

SUBROUTINE RDCARDS

PURPOSE:	To read the Simulator data from cards.
ENTRY POINTS:	RDCARDS
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AATTRIT, ABMDATA, DAMAGE, FUTPRINT, INBOMBF, INREP, IPRINT, IPSWICH, LATTRIT, NAMES, NSW, READ, TBMDATA, VULNDATA, ZONES
SUBROUTINES CALLED:	OPTPRIN, RANFSET*
CALLED BY:	SIMULATE

Method

The basic Simulator data are read and printed. R, the initial value of the random number generator, is read and printed. Subroutine RANFSET is called to set the random number generator. NET, the number of event tapes; IPRINT, the print option indicator; IPSWICH, the option flag for printing the INDATA blocks; IFUT, the option flag for printing the event execution times during the running of the game; and NSW, the option flag for printing event data at event execution time; are all read and printed. If NSW is nonzero, subroutine OPTPRIN is called to read data cards and initialize the printing sequence.

INBOMBF, which indicates whether unused History Table lines should be saved; INREP, which indicates whether reprogramming is to be conducted; FK, FCM, ALT, [illegible], and RPEN, area attrition parameters; the maximum number of zones; DEFHI, DEFLO, BKBRP, and PDLABT, local attrition parameters; are all read and printed. The vulnerability data are read but not printed. WTR21MT, maximum effective radius of a one-megaton weapon; DELAY(1) and DELAY(2), respectively, BLUE and RED delay times; and PTK(1) and PTK(2), respectively, BLUE and RED terminal kill probabilities; are all read and printed. IARDEF(1), PSEL(1), PAK(1), PREM(1), and PAKD(1), area ballistic missile parameters for side BLUE, are read and printed; and IARDEF(2), PSEL(2), PAK(2), PREM(2), and PAKD(2), for the RED side, are read and printed, as is H hour, the reference time of the game.

Subroutine RDCARDS is illustrated in figure 44.

---

*System Library Function

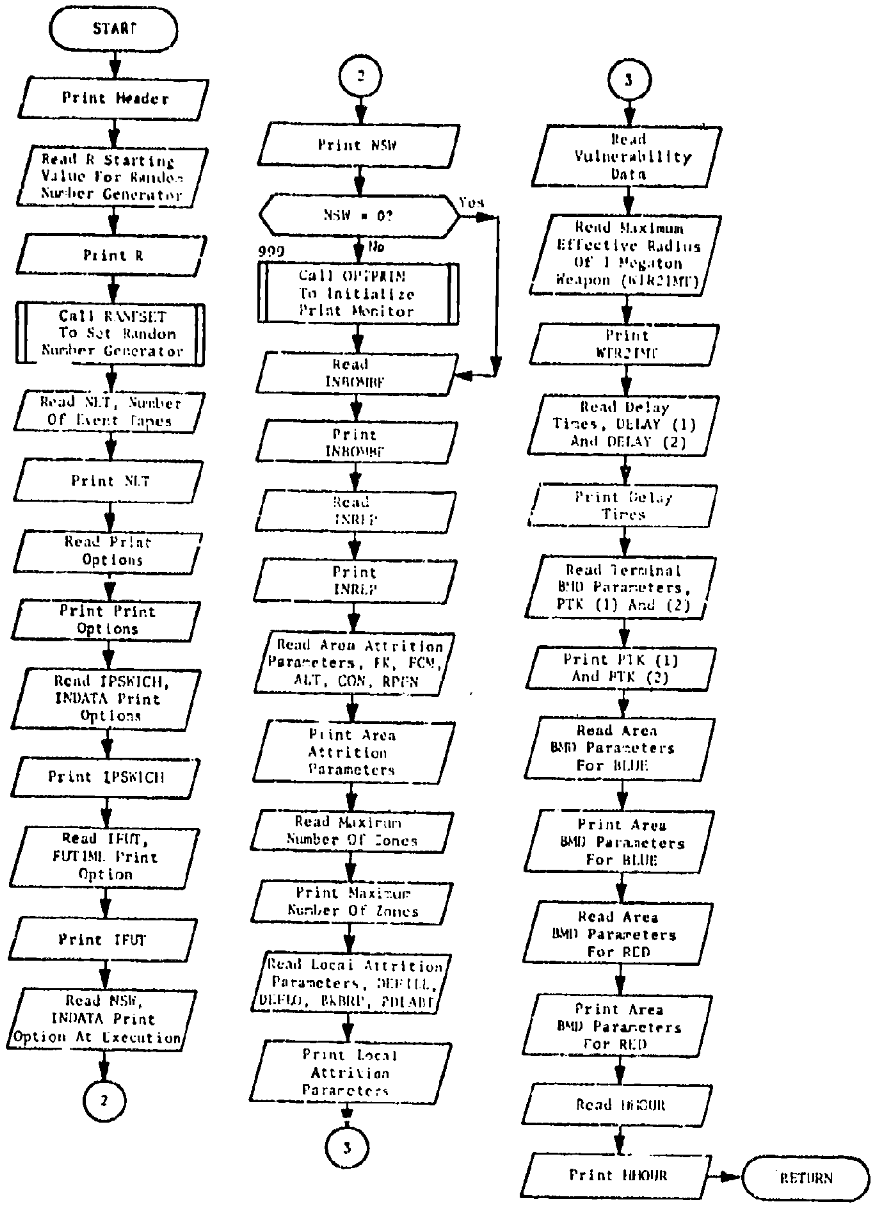


Fig. 44. Subroutine RDCARDS

SUBROUTINE READIN

PURPOSE:	To read data from the SIMTAPE tape into memory.
ENTRY POINTS:	READIN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	AREADAT, ASMS, BOMBER, BRKPNT, CAPACITY, COLAR, DAMAGE, FILABEL, MISSLE, NAVDATA, NCOL, PAYLOAD, STATUS, TANKER, TBMDATA, TWORD, WARHEAD, ZONES
SUBROUTINES CALLED:	ABORT, RDARRAY, RDWORD, SETREAD, TERMTAPE
CALLED BY:	SIMULATE

Method

Subroutine READIN calls SETREAD to put the library tape in read status. ITP, the current tape unit, is set to seven for the library tape in the calling subroutine.

If the tape label is incorrect, an error message is printed and the simulation is terminated. If the label is correct, then ITWORD, the number of items in the breakpoint arrays, is read in and stored. NTYPE is set to ITWORD and subroutine RDARRAY is called to read and store NTYPE values of INDBEGTY and of NAMETYPE. RDARRAY is called to read 15 values of each of the arrays NTYPECUM, NBLUETYP, INDBEGCL, and NAMCLS.

RDWORD is called to read and store in ITWORD the number of items in the hardness table. NVULN is set to ITWORD. RDARRAY is called to read store NVULN values of array VULN.

RDWORD is called to read and store in ITWORD the number of collocated targets. NCOL is set to ITWORD. If NCOL is greater than 4,000, a message is printed noting that array COLAR would be overflowed, and subroutine ABORT is called to terminate the simulation. Otherwise, RDARRAY is called to read and store NCOL items of array COLAR.

RDWORD is called to find NTDEF, and RDARRAY is called to read and store NTDEF items of array NTINT, the number of terminal defenders at each complex.

RDARRAY is called to read and store 60 items of the array AINT, the number of area interceptors of each area defense zone.

RDARRAY is called to read and store 20 items of the array NLRR, the number of long-range radars effective over each area defense zone.

RDARRAY is called to read and store 20 items of the array IOVERLAP, packed data giving the INDEXNO of a radar and the zones which it covers.

RDWORD is called to find MAXIND, the number of bases in the game. RDARRAY is called to read and store MAXIND items of the STATUS array.

RDWORD then finds the number of missile types NMIS, and the 11 missile arrays comprising the larger array MIS are filled by calls on RDARRAY.

RDWORD is called to read and store in ITWORD the number of items in each of the bomber arrays. NBOM is set to ITWORD. RDARRAY is called seven times to read and store NBOM items of each of the six bomber arrays comprising the larger array BOM.

RDWORD is called to read and store in ITWORD the number of items in each of the tanker arrays. NTANK is set to ITWORD. RDARRAY is called five times to read and store NTANK items of each of the four tanker arrays comprising the larger array TANK.

RDWORD is called to read and store in ITWORD the number of items in each of the ASM arrays. NASM is set to ITWORD. RDARRAY is called to read and store NASM items in each of PLABORT and CEPA.

RDWORD is called to read and store in ITWORD the number of items in each of the warhead arrays. NWHD is set to ITWORD. RDARRAY is called to read and store NWHD items each of PDUD, YIELD, and CEPW.

RDWORD is called to read and store in ITWORD the number of items in each of the zone arrays. NZON is set to ITWORD. RDARRAY is called to read and store NZON values each of AREA, ACCPOT, and ZDEFPOT.

RDWORD is called to read and store in ITWORD the number of items in the command/control effectiveness array. NTYPCC is set to ITWORD. RDARRAY is called to read and store NTYPCC items of array CCPOT.

RDWORD is called to read and store in ITWORD the number of items in the interceptor effectiveness array. NTYPIN is set to ITWORD. RDARRAY is called to read and store NTYPIN items of array DEFPOT.

RDWORD is called to read and store in ITWORD the number of items in each of the payload arrays. NPAYLOAD is set to ITWORD. RDARRAY is called five times to read and store NPAYLOAD items of each of the five arrays comprising the payload data. Then, data are read in describing the kill probability as a function of time which will be used in the analysis of naval attrition.

Subroutine TERMTAPE is called to terminate the tape, and the subroutine exits.

Subroutine READIN is illustrated in figure 45.

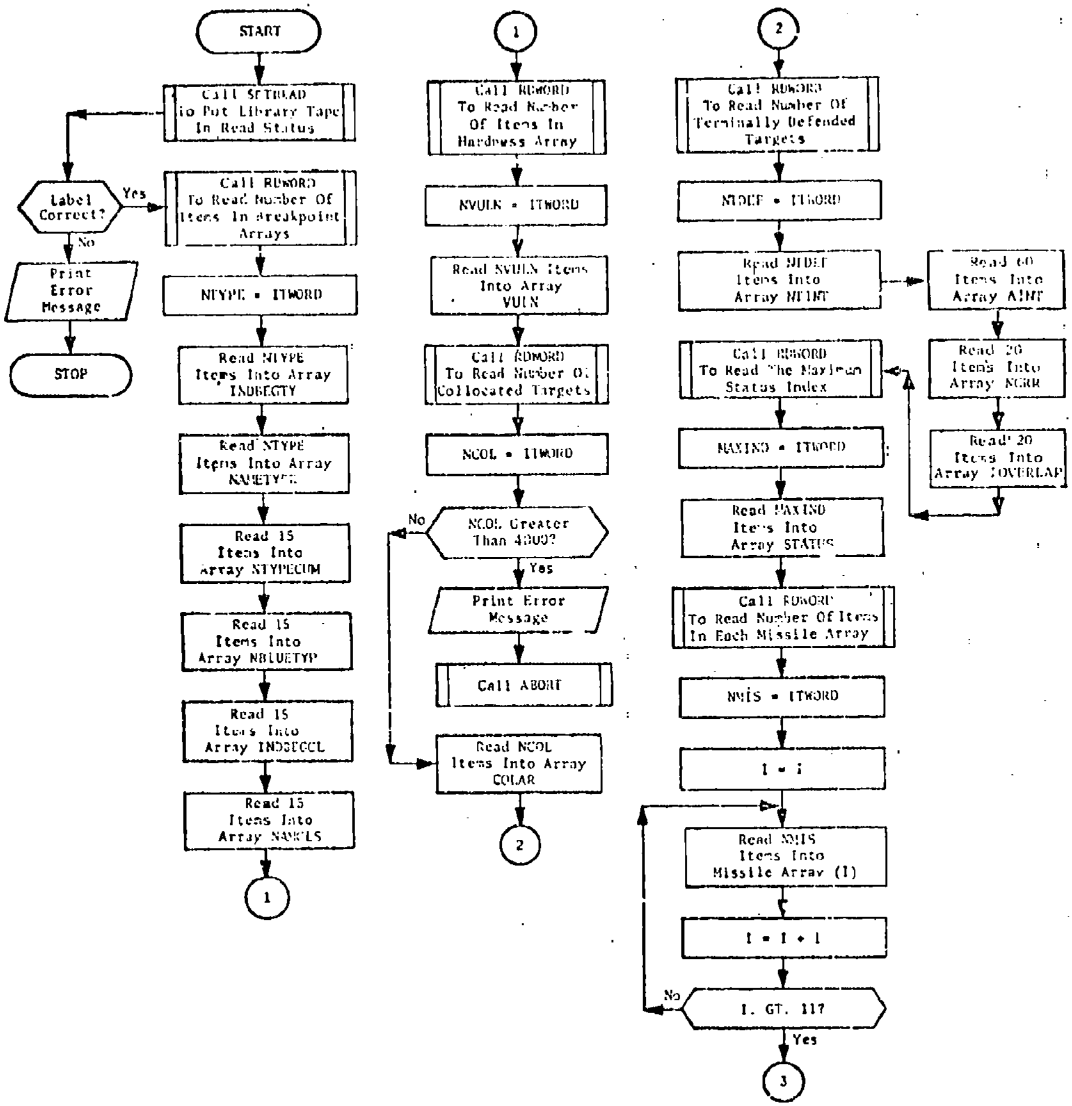


Fig. 45. Subroutine READIN
(Sheet 1 of 3)

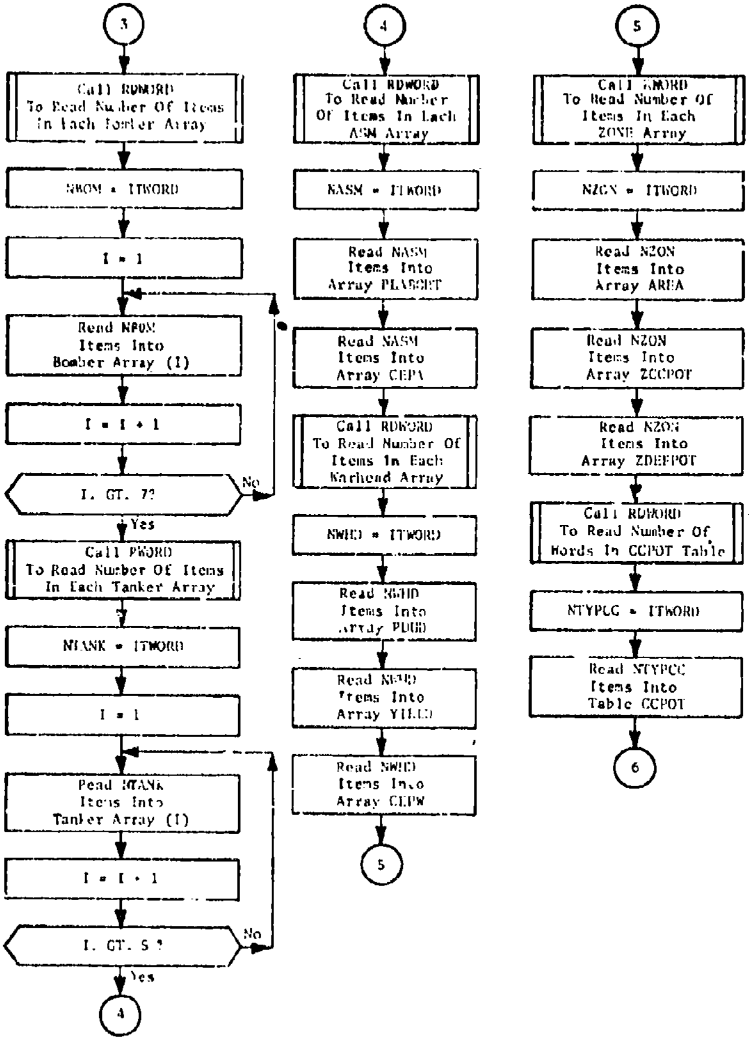


Fig. 45. (cont.)
(Sheet 2 of 3)

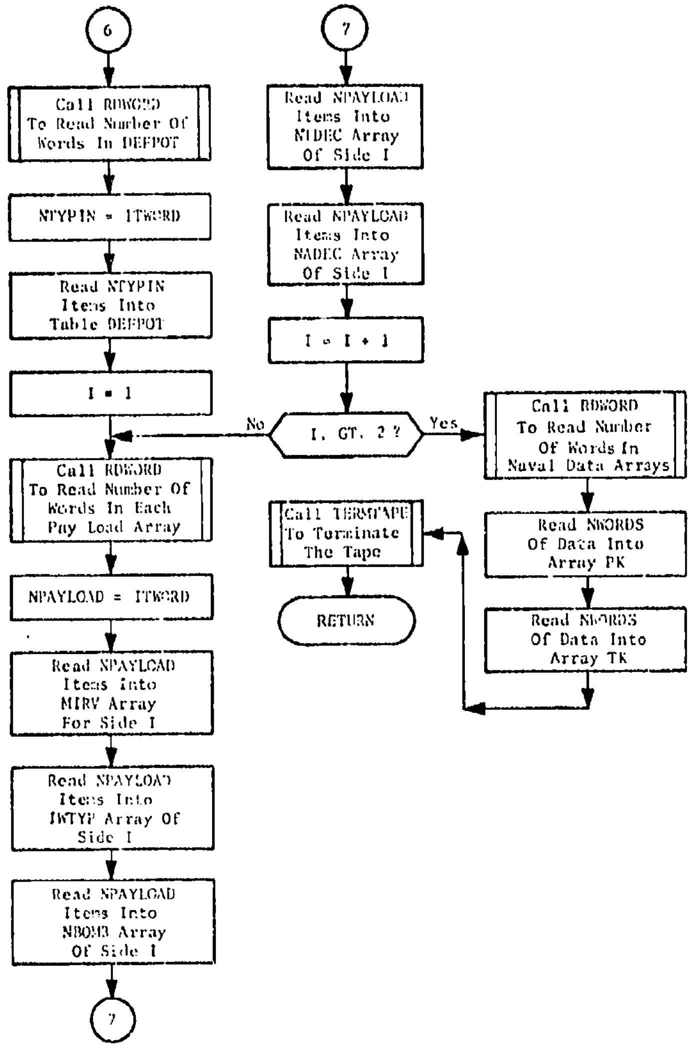


Fig. 45. (cont.)
(Sheet 3 of 3)

SUBROUTINE RECHEK

PURPOSE: To determine if the recovery base was killed after aircraft recovery.

ENTRY POINTS: RECHEK

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA, HISREC, KEYWORDS, NWORDOUT, RECOV, 19501

SUBROUTINES CALLED: HIST, IGET

CALLED BY: DCNEXT

Method

A test is made to see if the recovery is at the home base (event type 19 set by subroutine RECOVERY). If it is, IREC is set to the home base, stored in INDP. If not, IREC is determined by unpacking INDP and referring to the recovery array. In either case, the base is tested to see if it is alive. If it is, no record is made. However, if the base is dead, it is so recorded and the subroutine exits.

Subroutine RECHEK is illustrated in figure 46.

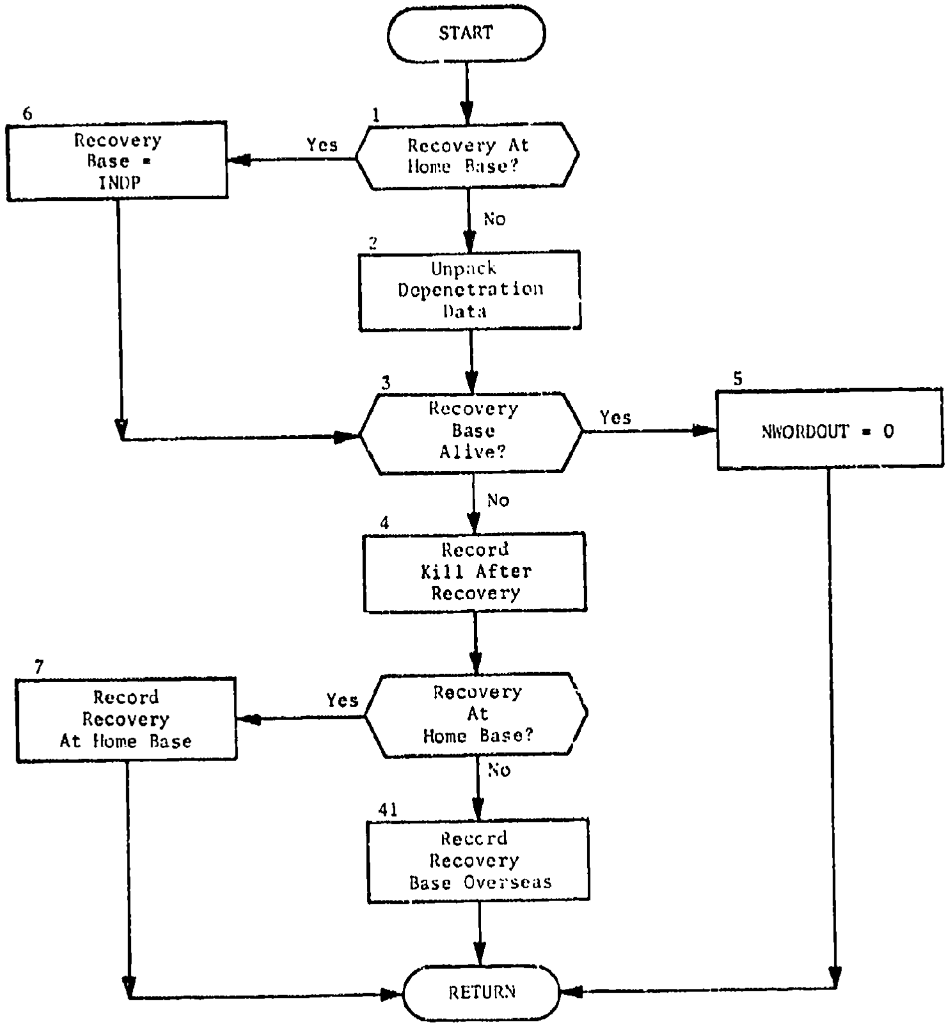


Fig. 46. Subroutine RECHEK

SUBROUTINE RECOVERY

PURPOSE:	To simulate the recovery of a bomber or tanker after completion of its mission.
ENTRY POINTS:	RECOVERY
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDATA, HISREC, HISTOUT, KEYWORDS, TIME, 19501
SUBROUTINES CALLED:	HIST, IGET, PLANTS
CALLED BY:	DONEXT

Method

The call to RECOVERY is recorded through a call to subroutine HIST.

If recovery is to be at the home base (INDE=16), a test is performed to see if it is still alive. If it is, a record is made and an event is planted to indicate that a later check should be made to determine whether the aircraft was killed after recovery. If the home base is dead, this fact is recorded and the subroutine exits.

If the recovery is associated with the depenetration corridor, the corridor and base indicator IR are unpacked from INDP. If IR=0, no live base was available when the recovery base was selected. This is recorded, and the subroutine exits. If a live base was available, the current status of this base is tested. If it is now dead, the fact is recorded and the subroutine exits.

A record is made of whether or not the base was saturated, a checking event is planted, and the subroutine exits.

Subroutine RECOVERY is illustrated in figure 47.

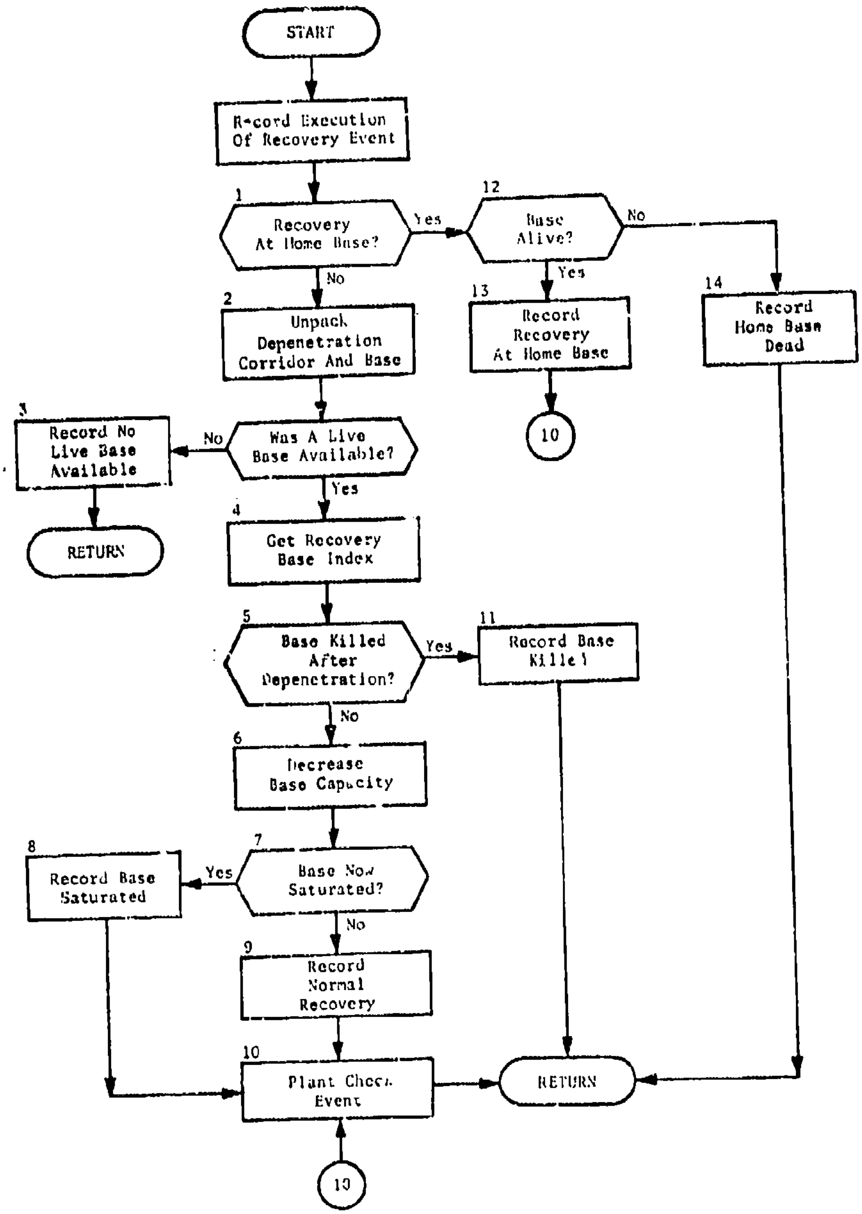


Fig. 47. Subroutine RECOVERY

SUBROUTINE REFUEL

PURPOSE: To simulate a bomber refueling.

ENTRY POINTS: REFUEL

FORMAL PARAMETERS: None

COMMON BLOCKS: BOMBER, EDATA, HISTOUT, HISTPEF, REFUEL, TIME

SUBROUTINES CALLED: HIST, NEXTEVNT, PLANTS, RANF*

CALLED BY: DONEXT

Method

The execution of REFUEL is recorded through a call to subroutine HIST. The number of weapons in the Weapon Table is temporarily set to zero. The nine right-most bits of INDP(IHT) are inserted into a local variable INDRA, the refuel area index, If the remainder of the word is zero, NT and NB (the numbers which specify the tanker-to-bomber ratio) are set equal to one. Otherwise, they are extracted by masking and shifting. The amount of fuel needed INEED is calculated to be 60 times the tanker/bomber ratio. If INDRA is nonzero, a test is made to determine whether the place is a refueling area. If NFTANK(INDRA) is greater than INEED, there is sufficient fuel in the area. If not, this is recorded and control is transferred as if there were a refueling abort.

If there are full tankers or if the place is not a refueling area, the probability of a refueling abort for this type of bomber PRABT(ITYPE) is compared to a random number generated by subroutine RANF to determine if there is a refueling abort.

If there is no refueling abort, the place is again tested to see if it is a refueling area. If so, the capacity of empty tankers in that area NETANK(INDRA) is increased by INEED, and the remaining fuel in the area is decreased by a like amount. If the next event is to be refueling, NEXTEVNT is called directly. Otherwise, the alternate events and weapons are discarded from the History and Weapon tables. NALT is set to zero, subroutine NEXTEVNT is called to plant the next bomber event, and the subroutine exits.

---

*System Library Function

If there is a refueling abort, this is recorded. If the next scheduled event is another refueling, a record is made of failure on the first of two refuelings, and a Recovery event is planted for the bomber at its home base. Otherwise, the History Table and Weapon Table pointers are reset to eliminate the primary plan, and the bomber proceeds with the alternate plan. If the next event is Recovery, the event is planted; otherwise, NEXTEVNT is called and the subroutine exits.

Subroutine REFUEL is illustrated in figure 48.

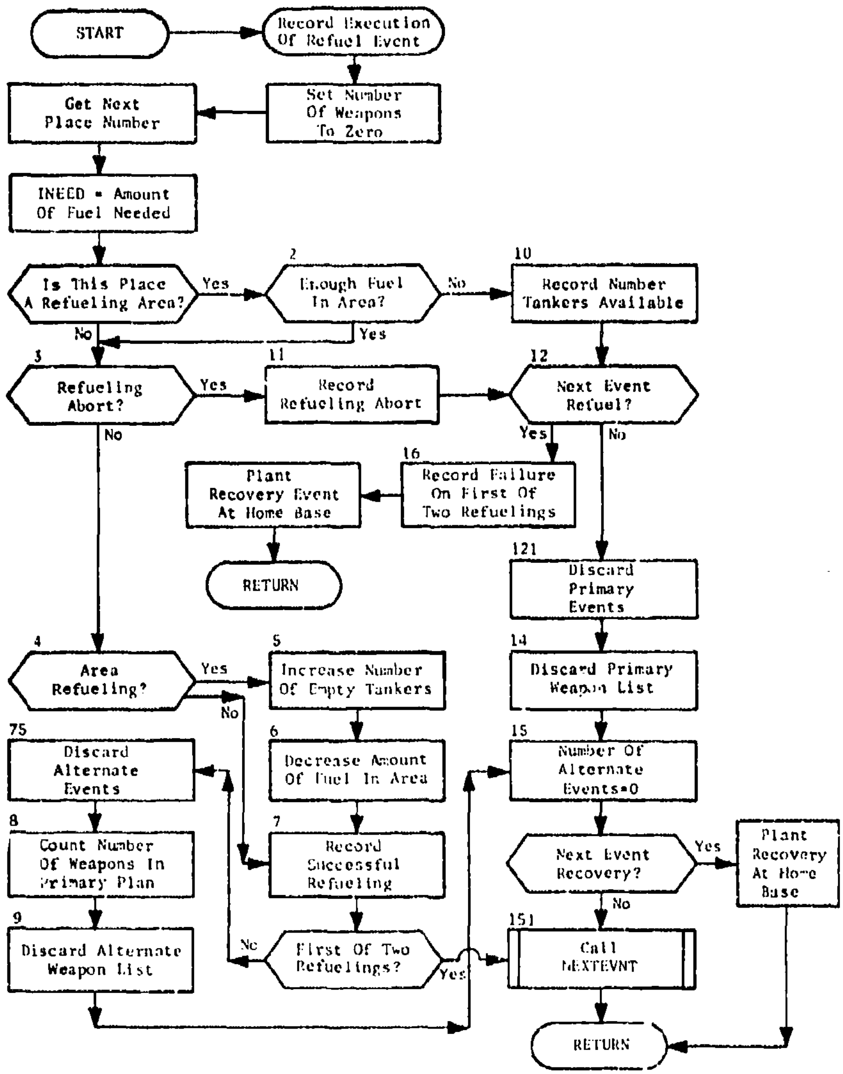


Fig. 48. Subroutine REFUEL

SUBROUTINE RETLM

PURPOSE:	To return the cells from an event in the event store back to available list memory.
ENTRY POINTS:	RETLM
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ESTOR, LISTMEM, 19501
SUBROUTINES CALLED:	None
CALLED BY:	DONEXT

Method

RETLM begins by finding the starting point of the event, which is contained in NEXTEV in common block /ESTOR/. The link to data, contained in the last word of the header cell, is moved up to the first word. The links to the data cells are then traced down until a link is contained which has the value zero, indicating there are no more data cells. The zero is replaced by IAVAIL, linking the event data cells into the beginning of available list memory. The address of the beginning of the header cell is put into IAVAIL and the subroutine exits.

Subroutine RETLM is illustrated in figure 49.

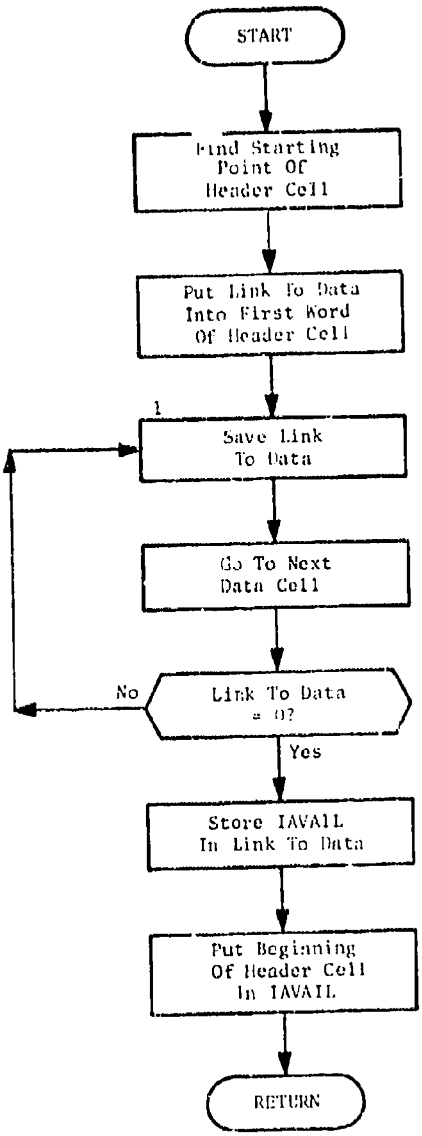


Fig. 49. Subroutine RETLM

FUNCTION RNDEV

PURPOSE:	To compute a random normal deviate.
ENTRY POINTS:	RNDEV
FORMAL PARAMETERS:	X - A dummy parameter
COMMON BLOCKS:	None
SUBROUTINES CALLED:	RANF
CALLED BY:	LATTRIT, TERMBMD

Method

RNDEV returns the sum of 12 random numbers minus 6.0. Function RNDEV is illustrated in figure 50.

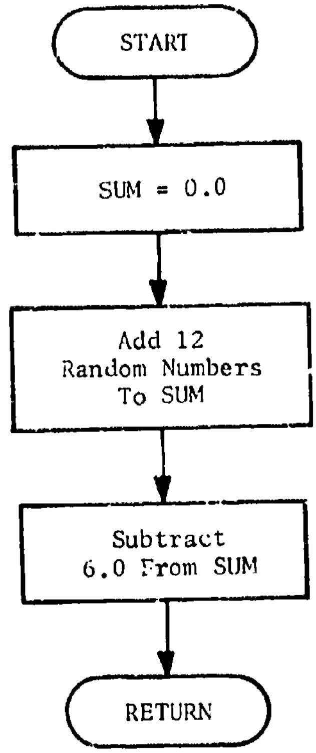


Fig. 50. Function RNDEV

SUBROUTINE SQUEEZE

PURPOSE:	To transfer data from list memory to array INDATAP.
ENTRY POINTS:	SQUEEZE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ESTOR, EVINDEX, 19501
SUBROUTINES CALLED:	ZABORT
CALLED BY:	DONEXT

Method

IPK, the pointer to the current element in INDATAP, is initialized to three. From the header cell of the list memory, the link to the next event is stored in INDEVB(1), and the link to the first data cell is stored in IX. The event time is stored in INDATAP(1), and in INDATAP(2) are packed the event format, the event type, and (just before exiting) the number of words of INDATAP used. This is done to provide for the situation in which the event is being spilled to an external file. If IX=0 (no data), the subroutine returns.

Four data words from a data cell are stored in successive cells of INDATAP as many times as necessary until either IX=0 (no more data, in which case the subroutine exits) or until there is danger that the bounds of INDATAP will be exceeded. In the latter case, an error message is printed out, and subroutine ZABORT is called to terminate the simulation.

Subroutine SQUEEZE is illustrated in figure 51.

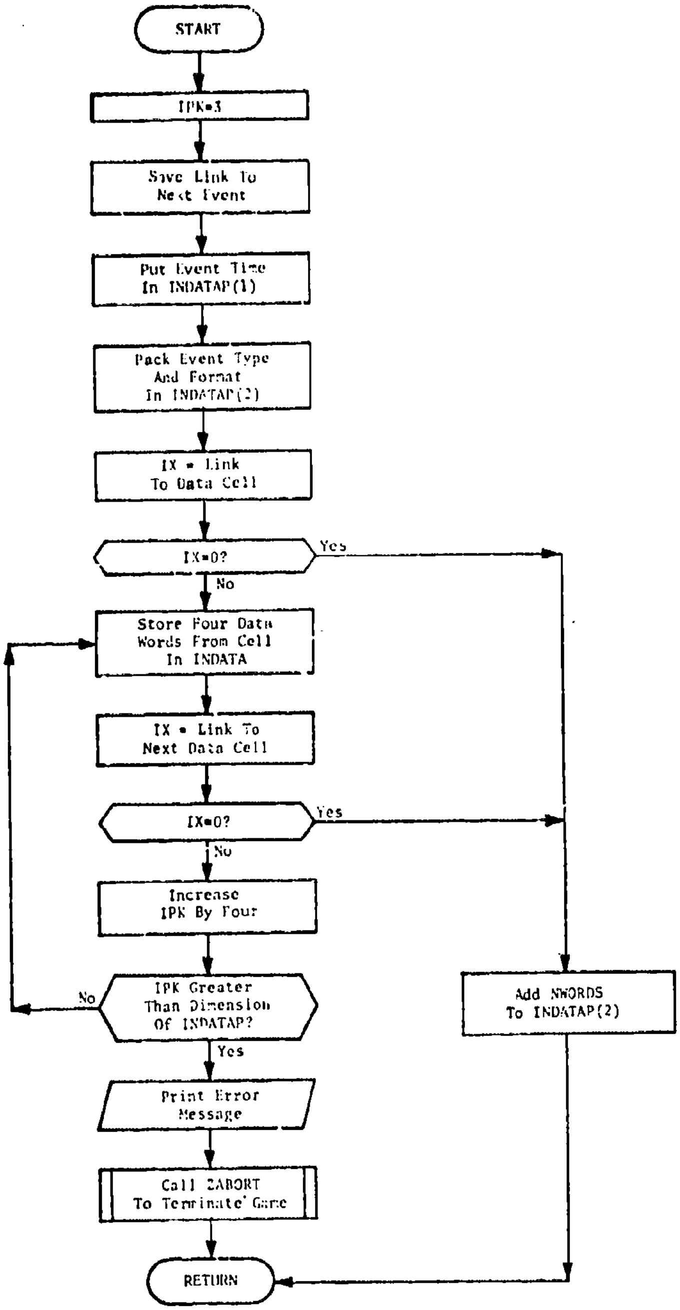


Fig. 51. Subroutine SQUEEZE

SUBROUTINE SSTAT

PURPOSE:	To compute and print zone status information.
ENTRY POINTS:	SSTAT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	HISTOUT, IPRINT, NAMES, NWORDOUT, TIME, ZONES, ZSTATUS
SUBROUTINES CALLED:	HISTWRIT, PLANTS
CALLED BY:	DONEXT, ENDGAME

Method

JT, the number of zone status events executed, is increased by one. The following computations are made for both of the two sides. NPENTOT, the total number of penetrators, is initialized to zero. IBEG is set to MINZONE(K), the minimum zone number for the current side. IEND is set to MAXZONE(K), the maximum zone number for the current side. NPENZ(I), the number of penetrators in each zone I from IBEG through IEND, are added together and stored in NPENTOT.

If NPENTOT is nonzero, current game time TIME and the name of the current side NAMESIDE(K) are printed. For each I, the zones from IBEG through IEND, the following information is printed: I, the zone number; NPENZ(I), the number of penetrators in the zone; ZDEFPOT(I), the defensive effectiveness in the zone; ZCCPOT(I), the command and control effectiveness in the zone; and KILLZ(1), the number killed in the zone by area attrition.

If NPENTOT is zero, no printout is made for the current side.

After the above has been executed for both sides, JT is compared to JTMAX, the desired number of zone status events to be executed. If JT is equal to JTMAX, the subroutine exits. Otherwise FUTIME, the time for the execution of the next zone status event, is set to TIME plus one-quarter hour. JOANNA, the second parameter in the call to subroutine PLANTS, is set so that PLANTS does not reference the format index arrays, which SSTAT does not use. Subroutine PLANTS is called to plant a zone status event for execution at FUTIME. NWORDOUT, the number of words of the HISTOUT block to be used, is set to zero, and the subroutine exits.

Subroutine SSTAT is illustrated in figure 52.

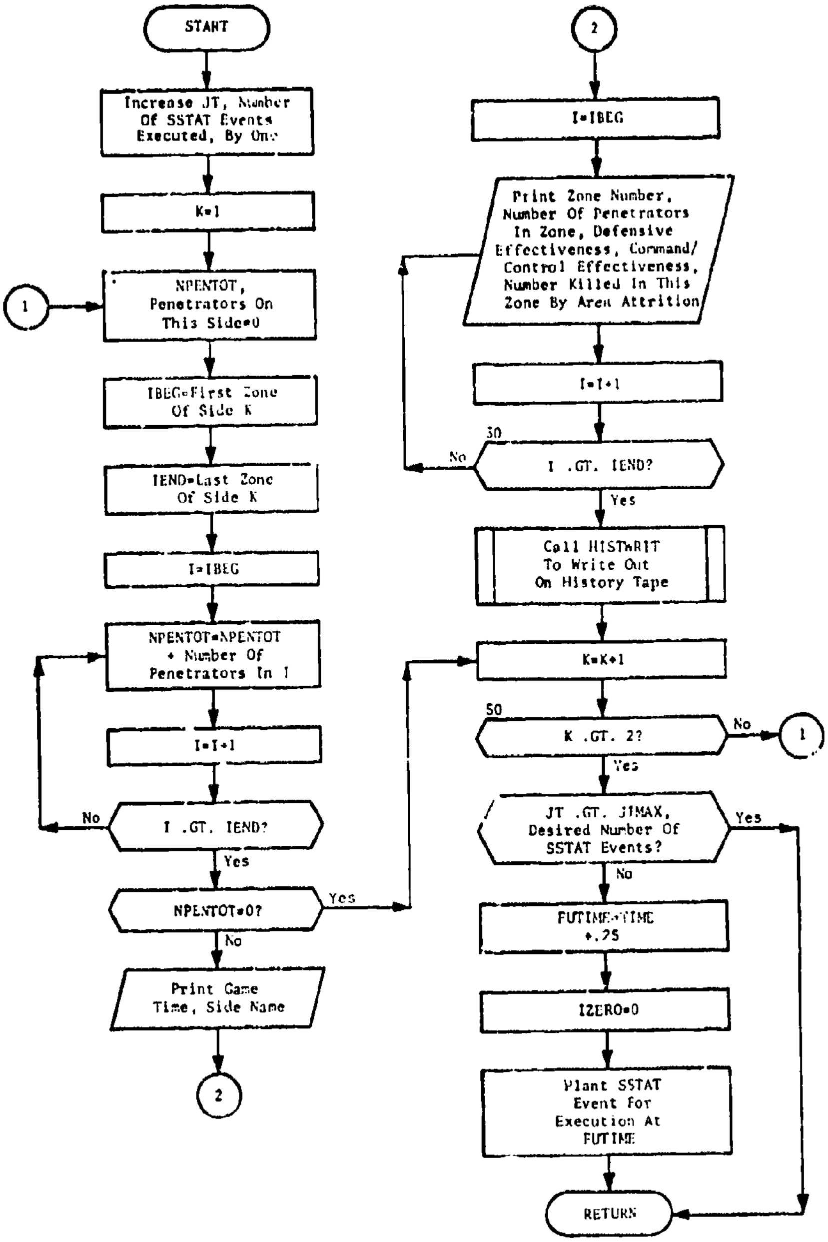


Fig. 52. Subroutine SSTAT

SUBROUTINE STATSUM

PURPOSE: To print a summary of the status of game items by class and type.

ENTRY POINTS: STATSUM

FORMAL PARAMETERS: None

COMMON BLOCKS: BRKPNT, KEYWORDS, NAMES, 19501

SUBROUTINES CALLED: IGET, PAGESKP

CALLED BY: ENDGAME

Method

A page is ejected on the standard output, and NTYPES is set to NTYPECUM(15), the total number of types in all classes.

For each type, the following computations are made. The number of survivors NSURV(I) is initialized to zero. IBEG is set to the first index of the current type and IEND to the last. Since all items were alive initially, NINIT(I), the number of items of the current type alive initially, is found from the two indices. Through successive calls to subroutine IGET, the status indicator in the STATUS array word for each item of the current type is tested. IGET returns a zero for a dead item and a one for a live item, so NSURV(I), the number of items of the current type surviving, is computed by adding together all the statuses.

NT, the number of types, is initialized to zero.

For each class, the following computations are made and results printed. The current class name NAMCLASS(K) is printed. For each side, the following is done. The current side name NAMESIDE(L) is printed. If the current side is side one, BLUE, IBEG is set to the number of types NT plus one. IEND is set to NT plus NBLUETYP(K), the number of Blue types in the current class. If the current side is side two, RED, IBEG is set to IEND plus one and IEND and NT are both set to NTYPECUM(K), the total number of types in the current class.

In either case, using IBEG and IEND as limits, the name of the current type NAMETYPE(I), the beginning base index for the type, and NINIT(I) are printed for all types. In addition, if the type is one for which alive-dead status is kept, then the number surviving, the number killed, and the percent killed are printed out. The subroutine then exits.

Subroutine STATSUM is illustrated in figure 53.

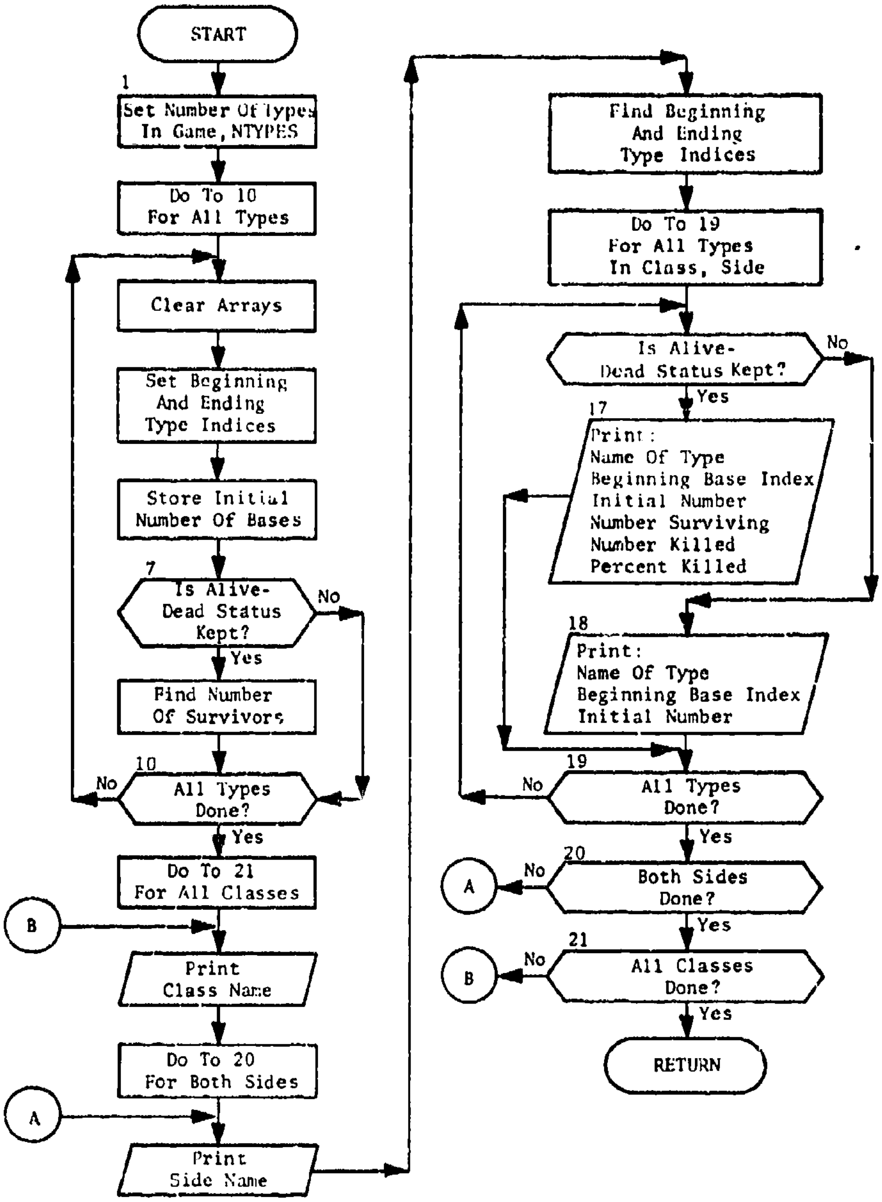


Fig. 53. Subroutine STATSUM

SUBROUTINE TERMBMD

PURPOSE: To test for survival of incoming warheads against a local defense system.

ENTRY POINTS: TERMBMD

FORMAL PARAMETERS: None

COMMON BLOCKS: EDATA, GROUND, HISTABM, INDEX, KEYWORDS, MISSLE, TBMDATA, WARHEAD, 19501

SUBROUTINES CALLED: BDAMAGE, HIST, IGET, LOGF,* RANF,* RNDEV

CALLED BY: AREABMD

Method

Subroutine HIST is called to record the call to TERMBMD. NPEN, NAL, and NDET are initialized to zero. SIDE, attacking side, is subtracted from three to give OSIDE, defending side. If there are no terminal aim points, the subroutine exits. ITERM, the terminal defense index for target INTAR, is retrieved through a call to IGET and stored in IT. If IT is zero, there are no interceptors, and NPEN, number of penetrators, is set to NWHDS, number of warheads, and control is transferred to statement 23, below.

If IT is nonzero, the target is defended. NTINT(IT), the number of terminal interceptors for this complex, is tested. If IT is zero, then NPEN is set to NWHDS and control transferred to statement 23.

If interceptors are available, enough are assigned to the incoming objects to achieve approximately a .99 kill probability considering PTK(OSIDE), probability of terminal kill for this side. The number of interceptors allocated NAL equals the number allocated per point N times TAIM number of terminal aim points. If NAL is not greater than NTINT(IT), the kill probability PK is found by subtracting from 1.0 the quantity 1.-PTK(OSIDE) raised to the Nth power. Control is then transferred to statement 17, below.

If NAL is greater than NTINT(IT), N is set to one, so only one interceptor is allocated per object. NAL is therefore set equal to TAIM, and the kill

*System Library Function

probability formula is reduced to PK=PTK(OSIDE). If there are not enough interceptors for N=1 (NAL greater than NTINT(IT)), control is transferred to statement 38, below.

Statement 17: NTINT(IT) is reduced by NAL. The following test is then made for each of NWHDS warheads. If PK, kill probability, is less than a random number generated by subroutine RANF, then NPEN, number of penetrators, is increased by one. When all NWHDS tests have been made, control is transferred to statement 23.

Statement 23: If NPEN is nonzero, warheads have leaked through. SIG, missile sigma, is calculated by multiplying the CEP for this missile type CEPM(ITYPE), times a constant; and IAGX and IAGY, actual ground zero components, are computed by multiplying SIG times RNDEV, a random deviate produced by function RNDEV. SIG is then set to the warhead CEP, CEPW(NWTYP), times a constant. The following check is made for NPEN warheads. If PDUD(NWTYP), dud probability for this type, is less than a random number, number detonated NDET is increased by one. When all NPEN tests have been made, control is transferred to statement 32.

Statement 32: HIST is called to record the results of the allocation. The following is done for each of the NDET detonating warheads. JAGX and JAGY, actual ground zero coordinates for this warhead, are each computed by multiplying SIG times RNDEV. The desired ground zero components for the next target are computed by increasing DGX by the sum of IAGX, and JAGX and DGY by the sum of IAGY and JAGY. Subroutine BDAMAGE is called for burst damage assessment. After this has been done NDET times, the subroutine exits.

Statement 38: There are insufficient interceptors for N=1. NAL, number allocated, is set to NTINT(IT), number of terminal interceptors, and NTINT(IT) is set to zero. The number of unattacked objects NREM is computed by subtracting NAL from TAIM. ICAT is initialized for randomizing. Words one through NWHDS are set to one and words NWHDS+1 through TAIM to zero. Subroutine REORDER is called to randomly reorder INDEX. Words INDEX(1) through INDEX(NREM) of ICAT are tested. If the value is one, the object is a penetrator, and NPEN is increased by one. After these NREM words have been tested, NWHDS is decreased by NPEN, and control is transferred to statement 18.

Subroutine TERMBMD is illustrated in figure 54.

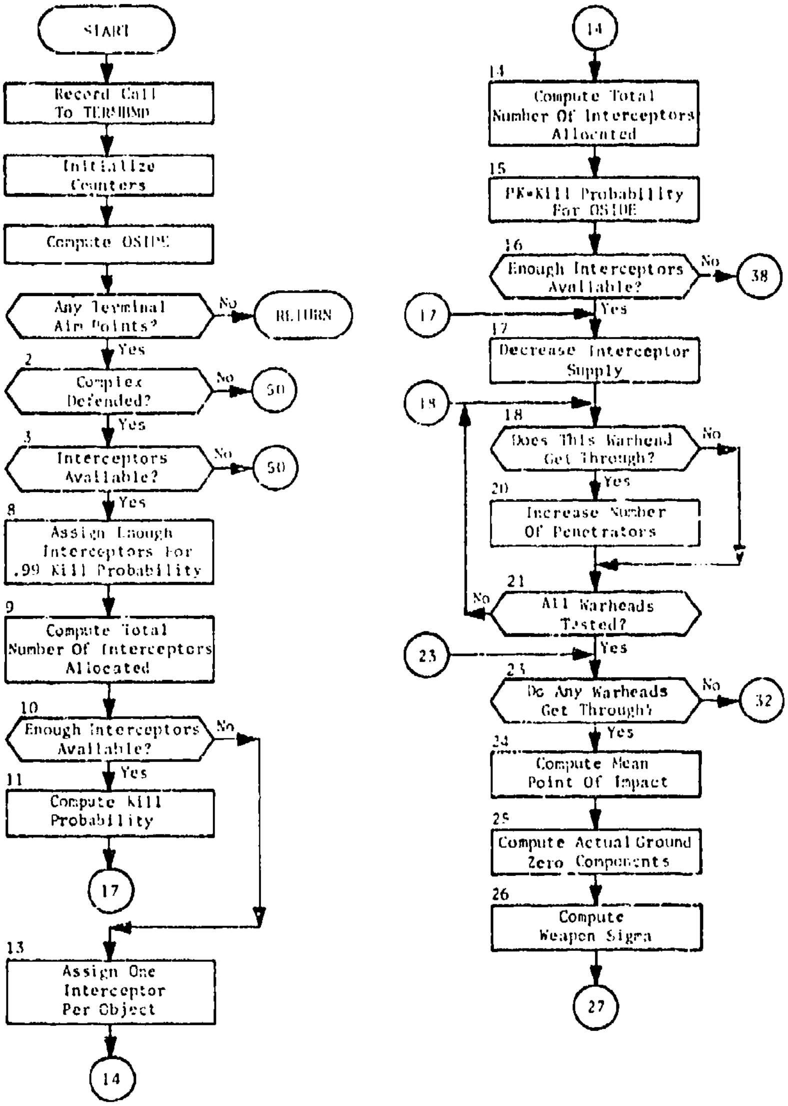


Fig. S4. Subroutine TERMBMD
(Sheet 1 of 2)

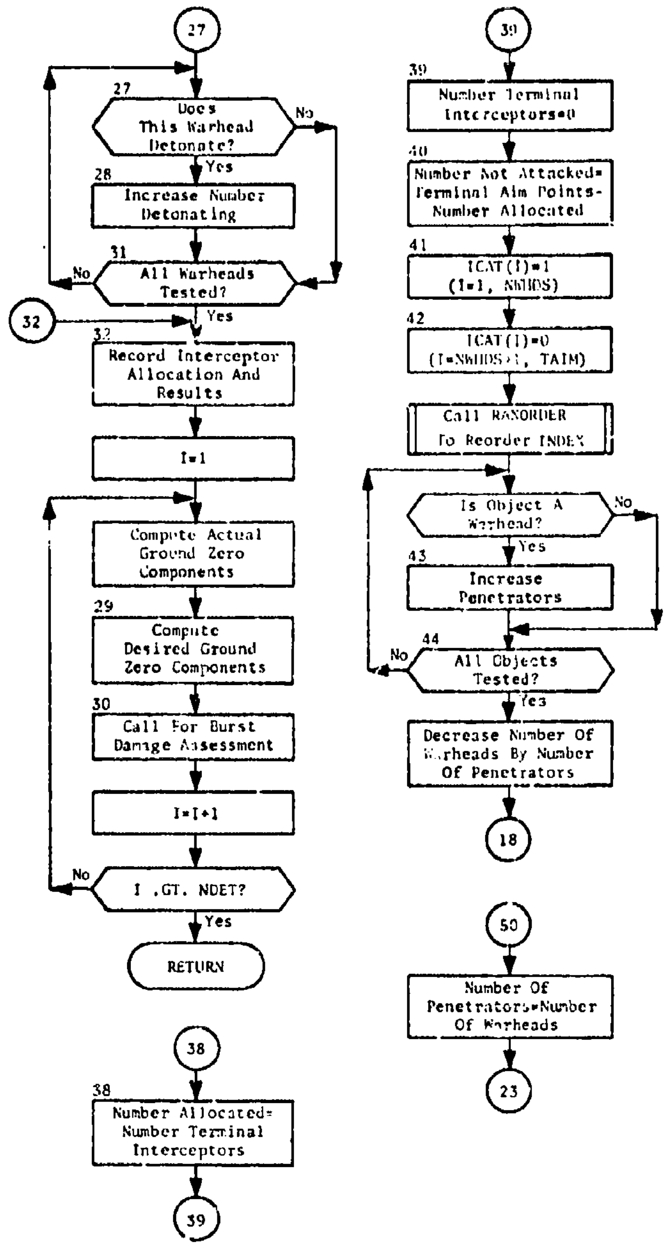


Fig. 54. (cont.)
(Sheet 2 of 2)

SUBROUTINE TRYLAUN

<u>PURPOSE:</u> To test for possible failures in launch of an individual missile.

<u>ENTRY POINTS:</u> TRYLAUN

<u>FORMAL PARAMETERS:</u> None

<u>COMMON BLOCKS:</u> EDATA, KEYWORDS, TRYL

<u>SUBROUTINES CALLED:</u> HIST, IGET, IPUT, RANF *

<u>CALLED BY:</u> MLAUN

Method

TRYLAUN is called by MLAUN. The number of missiles processed NCYCLE is incremented by one. The current missile index WPNLIST(NCYCLE) is stored in IND, and the current silo index LAUNLIST(NCYCLE) is stored in INBASE. The missile-in-commission flag IC is turned on. The probability that the current missile is in commission PINCOMM is compared to a random number generated by library subroutine RANF to determine if the missile is in commission. If so, the number in commission NCOM is incremented by one. If not, this is recorded and the in-commission flag turned off.

In either case, TSTAT, the status of the silo, is retrieved through a call to function IGET and stored in ISTAT. If the silo is dead, IC is tested. If the missile was in commission, the facts that the silo is dead and that the missile was in commission are recorded, and control is transferred to the test for availability of reprogramming. If the missile was not in commission, the facts that the silo is dead and that the missile was not in commission are recorded, and control is transferred to the test for availability of reprogramming.

If the silo is alive, IC is tested to see if the missile is in commission. If it is not, control is transferred to the test for availability of reprogramming.

If the missile is in commission, NALIVE, the number of missiles alive, is incremented by one. The probability of no abort for this missile PNOABORT is compared to a random number.

---

*System Library Function

If there is an abort, this is recorded and PDEST, the probability of destructive abort, is compared to a random number. If there is not a destructive abort, control is transferred to the test for availability of reprogramming. If the abort destroys the silo, this is recorded and the silo status set to "dead" through a call to subroutine IPUT. Control is then transferred to the test for availability of reprogramming.

If there is no launch abort, the number of missiles not aborted NNABT is incremented by one. PFLTFAIL, the probability of failure in powered flight, is compared to a random number. If the missile fails, this is recorded and the reprogramming index IREP tested for the availability of reprogramming after failure in powered flight. If this is available, the number of missiles to try later NLATER is incremented by one. In either case, control is transferred to the test for availability of reprogramming.

If the missile does not fail in powered flight, NSUCC, the number of missiles successfully launched, is incremented by one. The successful launch is recorded and the current entry in the list of successful launches INDEXWPN(NSUCC) is set to NCYCLE.

The test for the availability of reprogramming is then made and NTEST, the number of missiles for which reprogramming is unnecessary, is set. If IREP is one, all missiles processed in this squadron are in this category, and NTEST is set to this quantity; if IREP is two, all the missiles in commission are in this category; if IREP is four or five, all those that did not abort are in this category. Control is then returned to MLAUN.

Subroutine TRYLAUN is illustrated in figure 55.

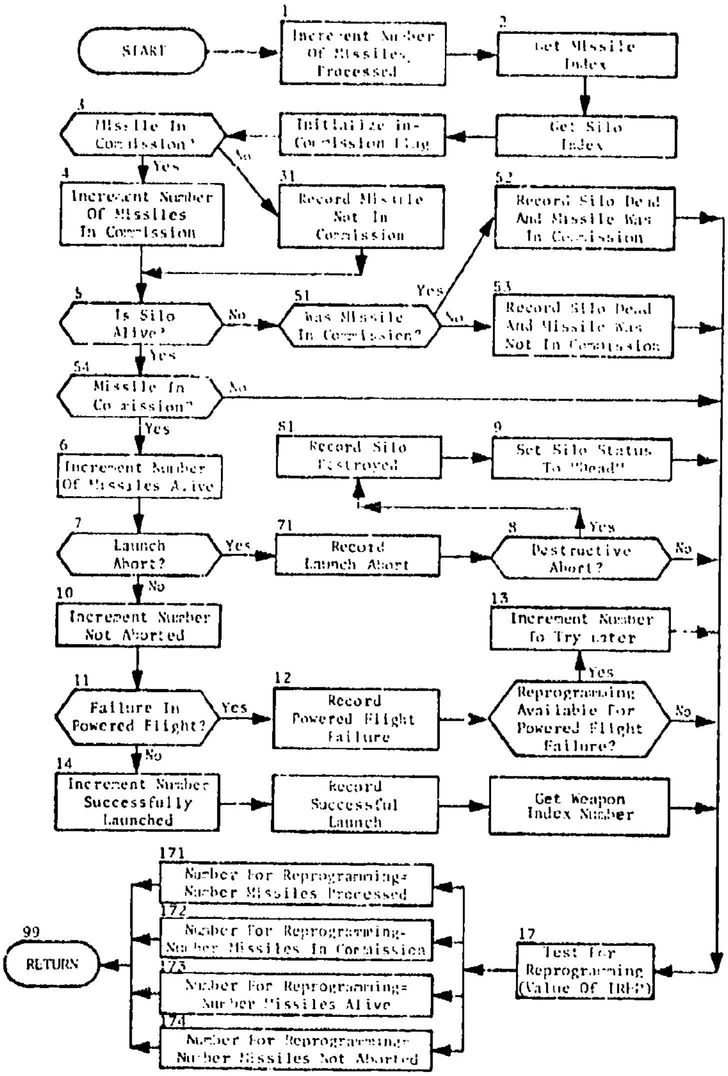


Fig. 55. Subroutine TRYLAUN

SUBROUTINE UNPFOR

PURPOSE:	To unpack the format index array into the array INDFORMO.
ENTRY POINTS:	UNPFOR
FORMAL PARAMETERS.	None
COMMON BLOCKS:	EPACK, EVENT, PLANTS
SUBROUTINES CALLED:	None
CALLED BY:	EVPACK, EVUNPK

Method

The array INDFOR is divided into 10-word blocks. Each block contains packed data words which control the packing and unpacking (from OUTDATA to OUTDATAP in the case of subroutine EVPACK; from INDATAP to INDATA in the case of subroutine EVUNPK). The value of INDF in common /EVENT/ determines which 10-word block is to be referred to. Each word of INDFOR is divided into eight six-bit segments. The first segment of the first word in the block contains the number of segments used for that value of INDF; the remaining segments contain the unpacking formats (indices to the array JFORMAT in common /FORMAT/).

The subroutine starts by testing if INDF is the same as the previous value. If it is, the array INDFORMO is already properly configured, and the subroutine exits. If not, counters are initialized, and the unpacking proceeds one word of INDFOR at a time. INDF is then saved and the subroutine exits.

Subroutine UNPFOR is illustrated in figure 56.

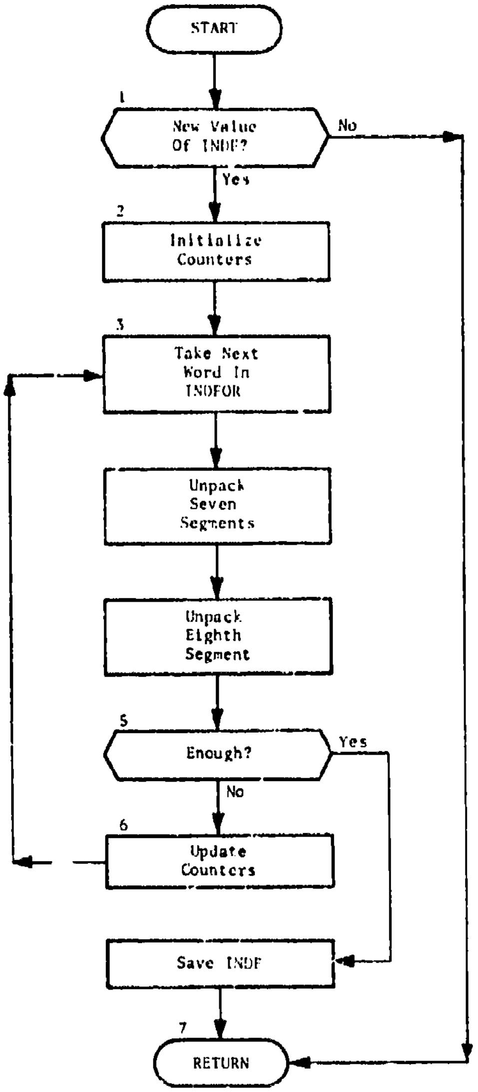


Fig. 56. Subroutine UNPFOR

SUBROUTINE UNSQUEEZ

PURPOSE:	To transfer data from array OUTDATAP to list memory.
ENTRY POINTS:	UNSQUEEZ
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ESTOR, LISTMEM, 19501
SUBROUTINES CALLED:	ERROL, EVSPILL
CALLED BY:	PLANT

Method

After initializing various counters, a test is made to see if there are any data to transfer. If not, the subroutine returns. If there are, a test is made to see if all elements of OUTDATAP have been transferred. When they have, the list is terminated by putting a zero into the "link to next data cell." If more elements are to be transferred, this is done one cell (four data words) at a time. If at any time before all are transferred it is found that no more list memory is available, subroutine EVSPILL (an entry to DONEXT) is called to spill the list memory onto an external file. The current event is then linked to the now empty list memory. If it is found necessary to call EVSPILL a second time, subroutine ERROL is called to print a diagnostic and terminate the simulation.

Subroutine UNSQUEEZ is illustrated in figure 57.

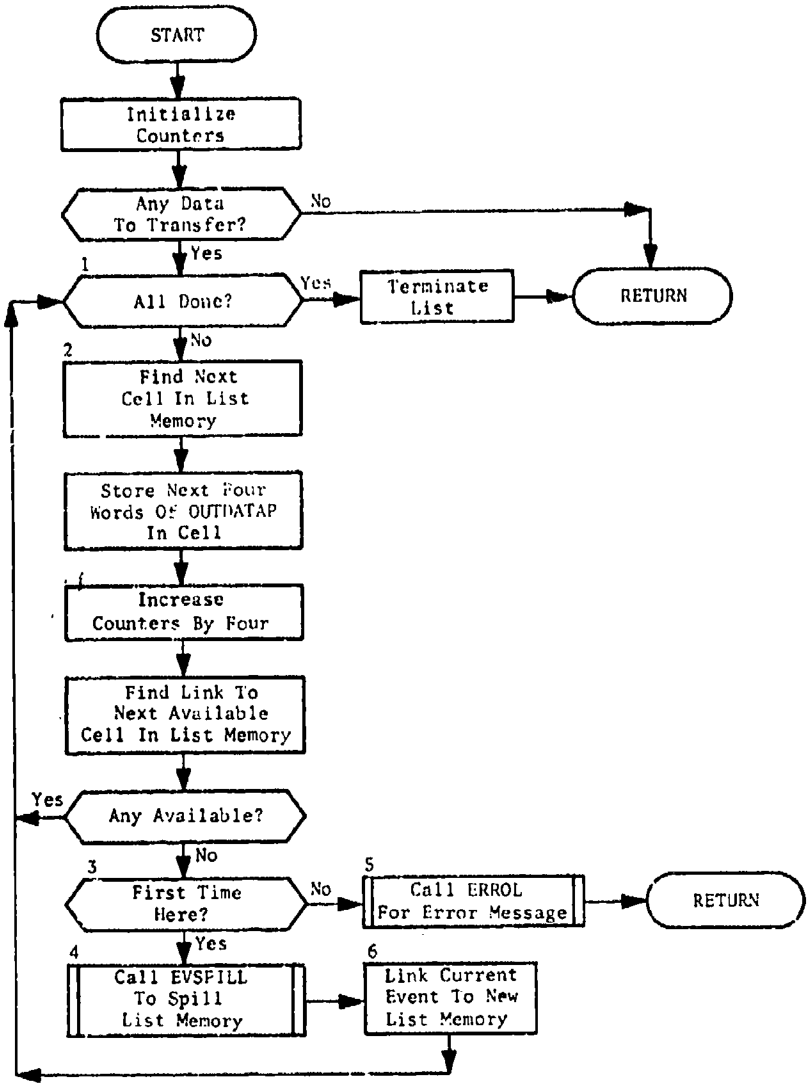


Fig. 57. Subroutine UNSQUEEZ

SUBROUTINE UPDIR

PURPOSE:	To update the directory of entry points to the sublists within the event store.
ENTRY POINTS:	UPDIR
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ESTOR, EVINDX, NEVTOT, 12501
SUBROUTINES CALLED:	None
CALLED BY:	DONEXT, PLANT

Method

The event store is divided into 10 sublists in order to be able to plant a new event efficiently, rather than searching through the entire event store to find the proper place to insert the list. The array EVTIME contains the maximum times of each list, and the array INDEVB contains the locations of the beginnings of each sublist in list memory. Both arrays are in common block /EVINDX/.

Subroutine UPDIR begins by linking all the sublists together. If the total number of events is no more than 50, only one sublist is used. If there are more, then one-tenth of the events are put in each sublist. This is done by running down the chain of events, and breaking it into sublists of appropriate size.

Subroutine UPDIR is illustrated in figure 58.

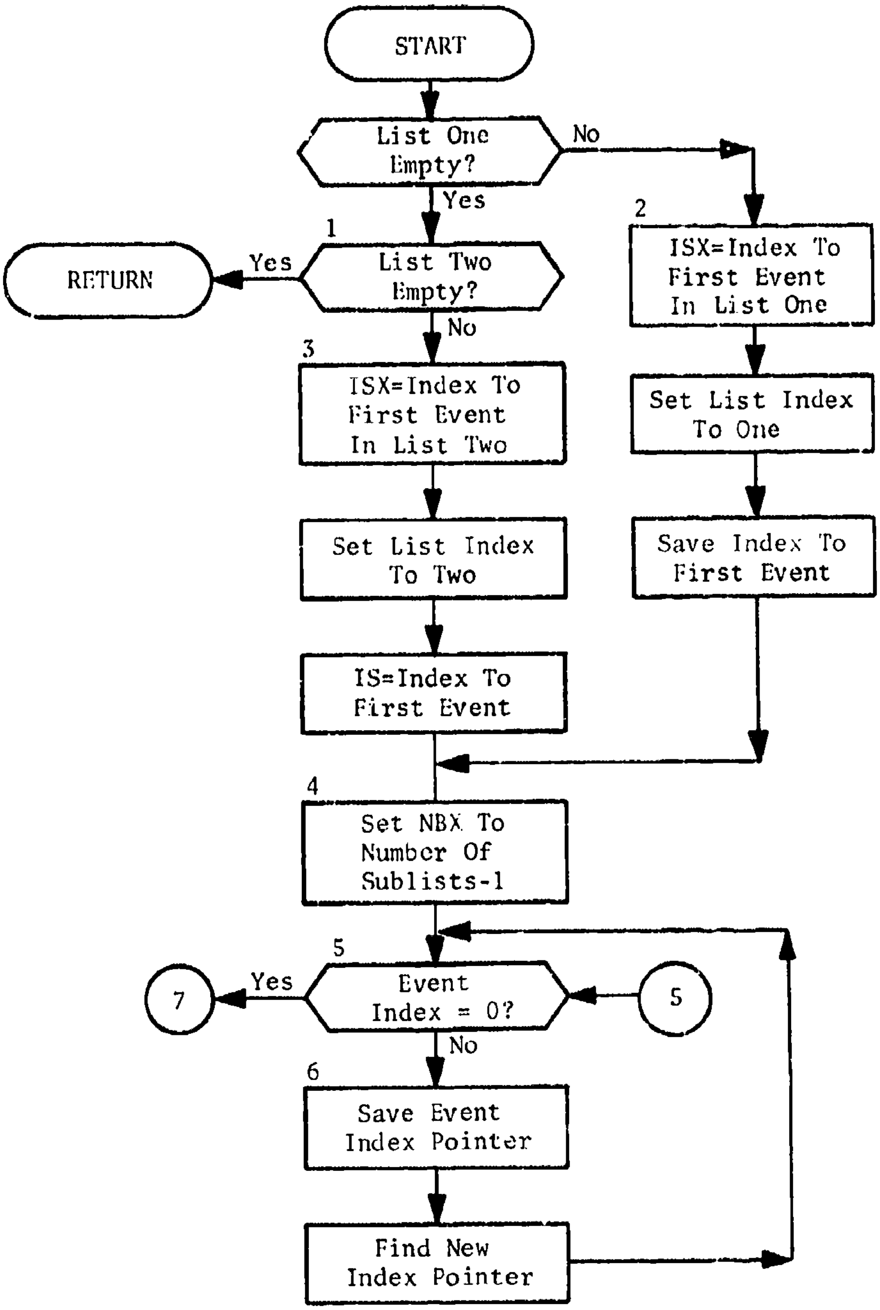


Fig. 58. Subroutine UPDIR
(Sheet 1 of 4)

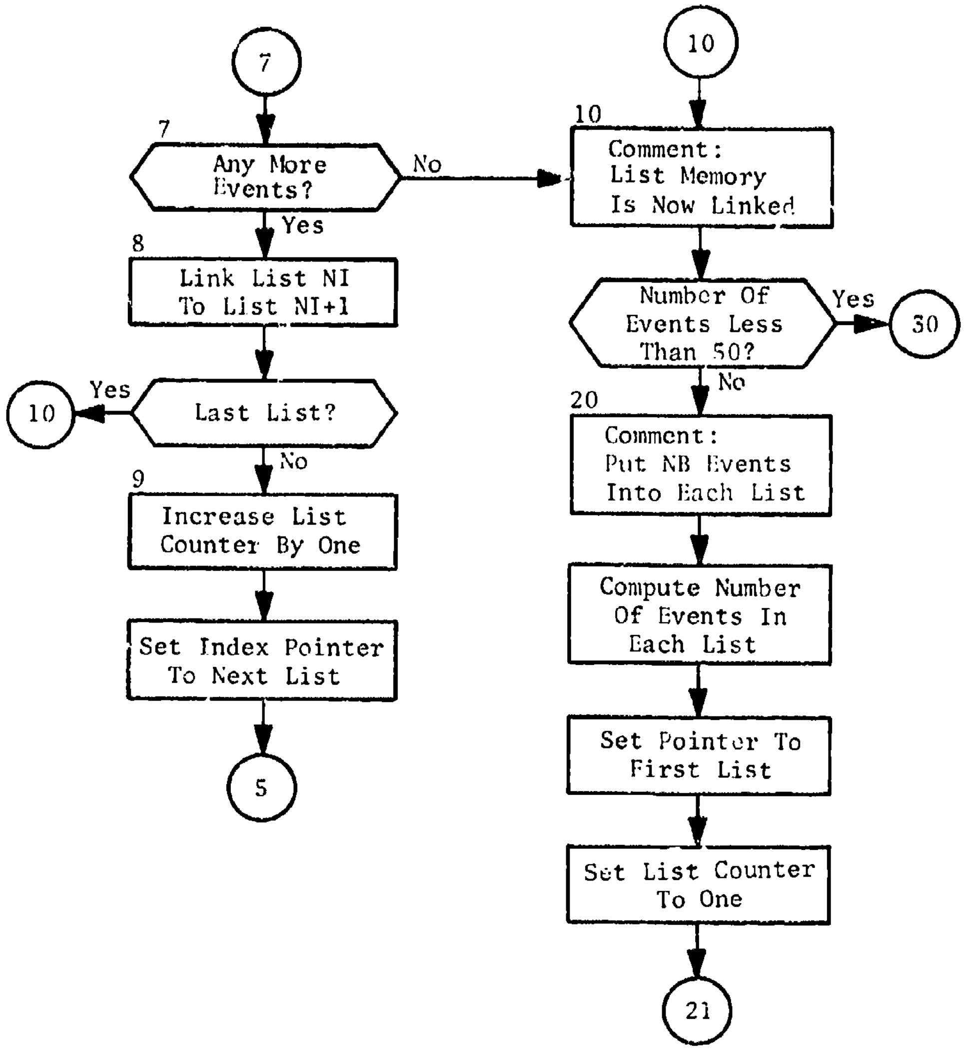


Fig. 58. (cont.)
(Sheet 2 of 4)

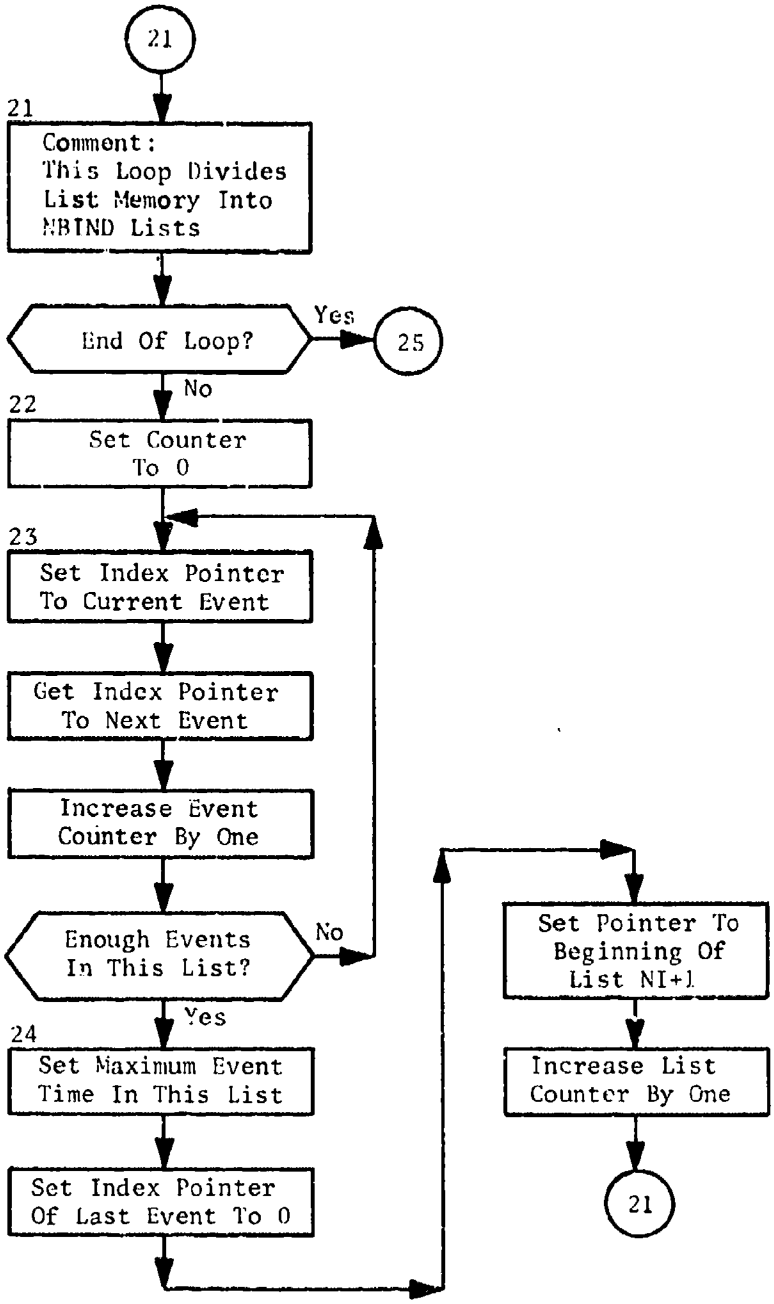


Fig. 58. (cont.)
(Sheet 3 of 4)

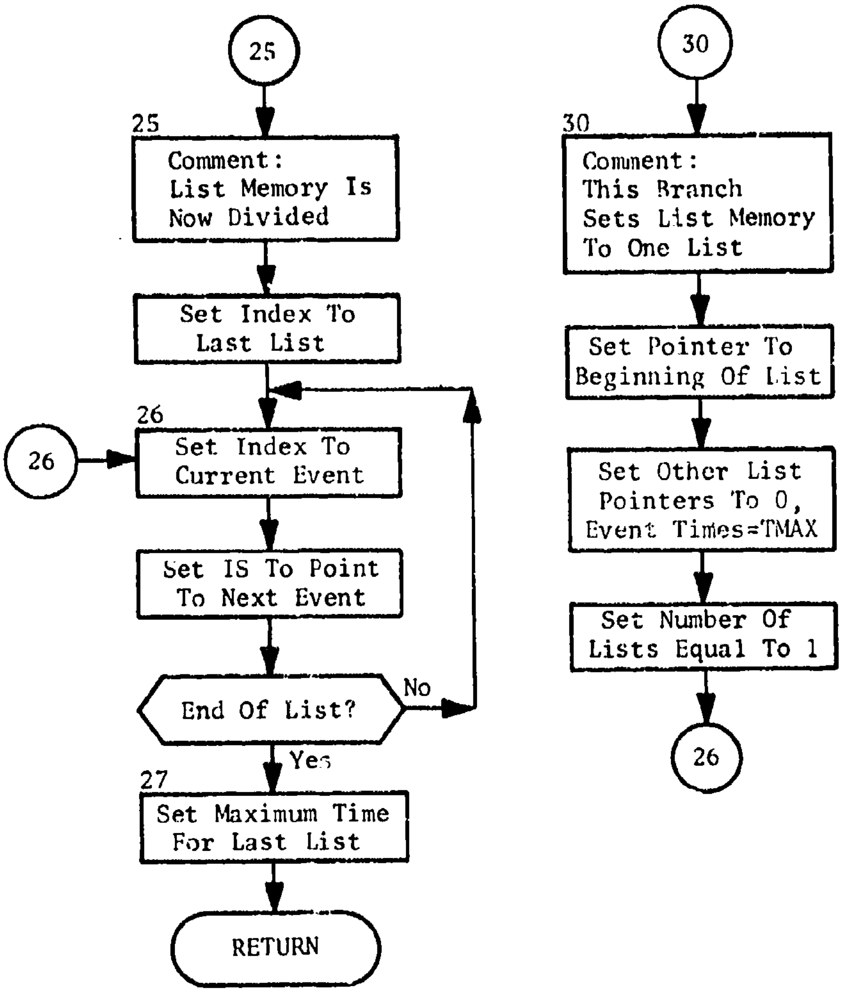


Fig. 58. (cont.)
(Sheet 4 of 4)

SUBROUTINE VLRAD

PURPOSE:	To find the lethal radius of a delivered warhead against its target.
ENTRY POINTS:	VLRAD
FORMAL PARAMETERS:	YIELD - Yield of weapon in megatons NVN - Vulnerability parameter of target HOB - Weapon height-of-burst index FN - Parameter specifying shape of damage function
COMMON BLOCKS:	VULNDATA
SUBROUTINES CALLED:	ABSF,* EXPF*
CALLED BY:	BDAMAGE

Method

NVN is decoded into the appropriate vulnerability number VN, the letter (P or Q), and the K-factor XK. The cube root of the yield is extracted. Then the adjusted vulnerability number AVN is determined by methods described in "Computer Computation of Weapon Radius," B-139-61, Air Force Intelligence Center. FN is set to 6. or 3. for P and Q type targets, respectively.

Common block /VULNDATA/ contains four arrays (for the four combinations of P or Q vulnerability and air- or surface-burst), each of which contains the natural logarithm of the lethal radius (in nautical miles) of a one-megaton burst. The data are at intervals of five vulnerability numbers. Subroutine VLRAD interpolates in the appropriate array to find the logarithm of the one-megaton lethal radius for AVN. The lethal radius of the weapon is then determined by exponentiating and multiplying by the cube root of the yield.

Subroutine VLRAD is illustrated in figure 59.

---

*System Library Function

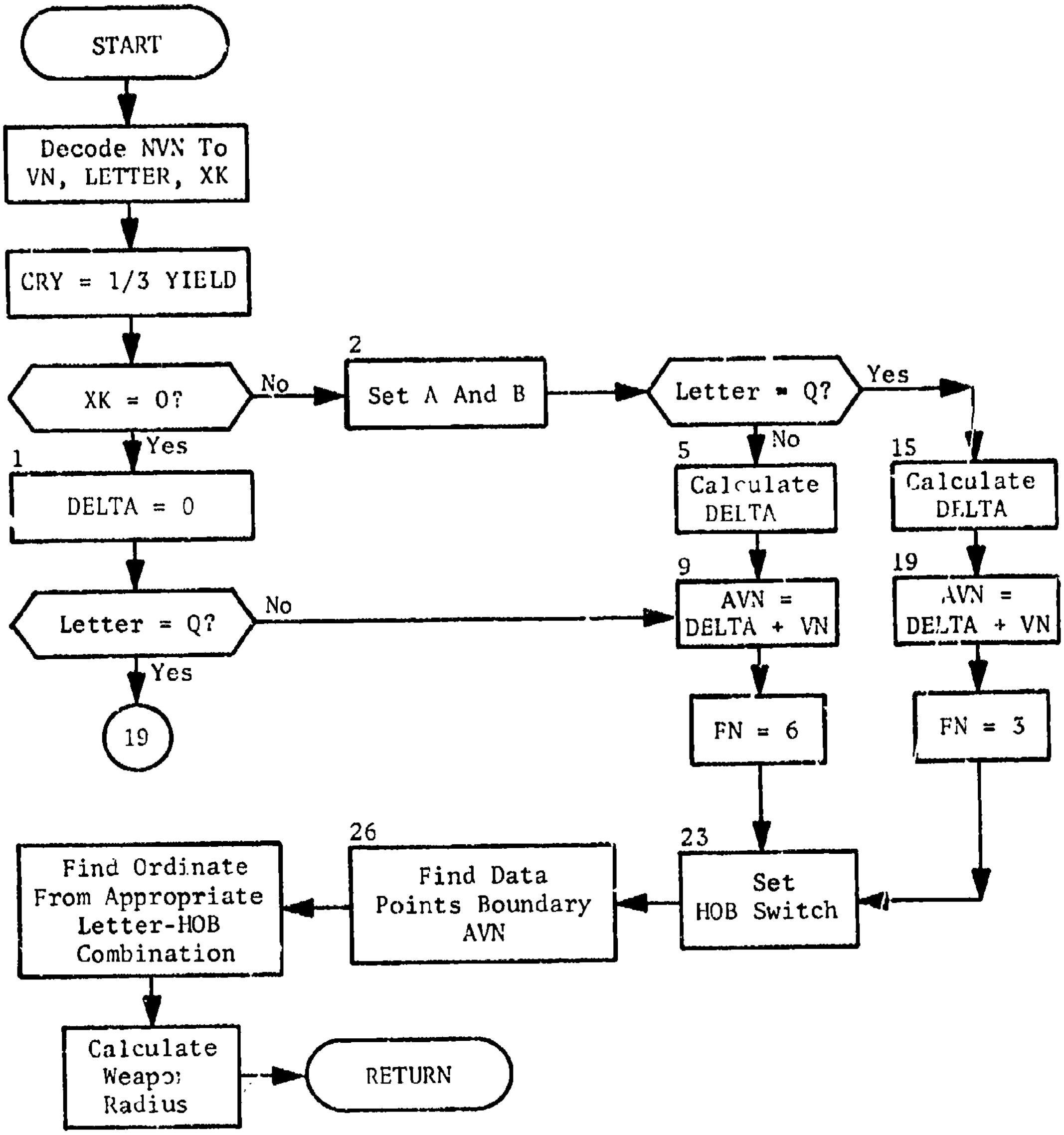


Fig. 59. Subroutine VLRAD

SUBROUTINE ZABORT

PURPOSE:	To terminate the History tape properly before calling subroutine ABORT.
ENTRY POINTS:	ZABORT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BRKPNT, ITP, MYIDENT, RECOV, TWORD, WARHEAD
SUBROUTINES CALLED:	ABORT, SETREAD, SKIPFILE, TERMTAPE, WRARRAY, WRWORD
CALLED BY:	DONEXT, EVPACK, EVUNPK, SQUEEZE

Method

The word 4HLAST, and the recovery base index array, are written on the History tape. The tape is then terminated, the breakpoint and yield tables added at the end, and ABORT is called.

Subroutine ZABORT is illustrated in figure 60.

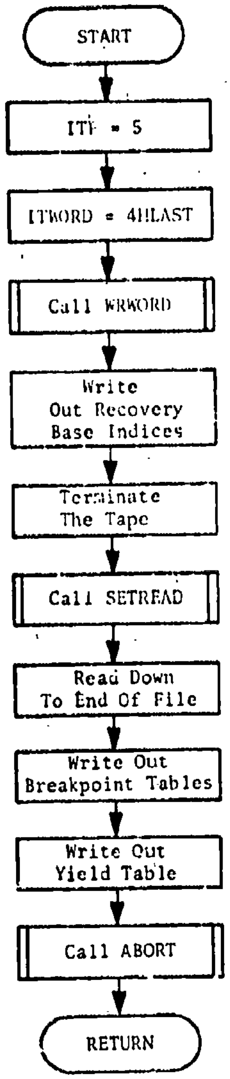


Fig. 60. Subroutine ZABORT

CHAPTER 3
PROGRAM READSUM

PURPOSE

READSUM performs three major tasks within the Data Output subsystem. The primary one is to expand the abbreviated output from SIMULATE using information gathered from the output of INDEXER, put the values of all pertinent attributes into standard attribute-value data base format, and write both a full History tape and one containing only Burst/Damage events.

The second function is to produce standard output summaries of bomber, missile, and tanker data for each side and the cumulative number of both Actual (simulated) Ground Zeros (AGZs) and megatons delivered for user-specified times, regions, and groups of countries. The data are stored, as the output from SIMULATE is being processed, and are printed at the conclusion of this processing.

The third task is to combine the Burst/Damage events from SIMULATE with information such as latitude, longitude, target designator, and task obtained from the output of INDEXER and to write four AGZ tapes: two in a format appropriate for input to the SIDAC damage assessment model and program WPNSUM; and two in a format usable in REST and OUTBLUE. SIDAC, WPNSUM, REST, and OUTBLUE are damage assessment systems maintained by NMCSSC. They are not part of the QUICK system.

INPUT FILES

The input to program READSUM comes from three sources: the HISTAPE produced by program SIMULATE, the INDEXDB tape produced by program INDEXER, and the user-input parameters.

The HISTAPE is made up of pairs of records. The first record contains the number of words in the second record. The second record contains the appropriate portions of an array called HISTOUT in program READSUM which was NHISTOUT in SIMULATE. The NHISTOUT records describe in an abbreviated form each event which occurs in SIMULATE.

The INDEXDB tape has breakpoint tables which give beginning indices of each class and type, and the number of Red and Blue types in each class. With their aid, it is possible to obtain the class, type, and side of any weapon from its index number. INDEXDB proper, which is an indexed data base tape, is read item by item, and information such as latitude, longitude, and target designator are stored by index number for later use in producing the output files. Weapon and payload information are also stored for later use.

The user-input parameters define the times, regions, and country groups for which standard summaries are to be stored and printed. In addition, they control the production and printing of the strike tapes.

OUTPUT FILES

Six files are output from program READSUM: HISTDATA, HISTAGZ, AGZREDRO, AGZREDSW, AGZBLURO, and AGZBLUSW.

Complete Event History Tape (HISTDATA)

This tape is written in attribute-value format and holds a record of each event for each vehicle in the war game. It is a combination of the partially coded data contained on HISTAPE and the information retrieved from INDEXDB. The attributes filled for each event type are enumerated in the descriptions of ADDATA, BDAMX, BOMBANDT, BOMBFX, CLAUNX, MLAUNX, and MATTRITX.

AGZ (Burst Event) History Tape (HISTAGZ)

This tape is a subset of HISTDATA containing only actual ground zero (AGZ) or burst event data. The attributes filled are listed in the discussion of BDAMX.

AGZREDSW, AGZREDRO, AGZBLUSW, and AGZBLURO

The AGZ (actual ground zero) tapes contain information on weapons which were simulated as having reached the target. Four card image tapes are prepared: two (table 18) for input to the SIDAC damage assessment model and program WPNSUM; and two (table 19) for input to non-QUICK programs REST and OUTBLUE.

CONCEPT OF OPERATION

Each event which has been simulated by SIMULATE is contained on HISTAPE in an abbreviated format. In order for these events to be summarized by the general-purpose program TABGEN, they must be translated into attribute-value format and augmented with other data not available on the HISTAPE. This is done by taking from /HISTOUT/ any data that can be directly transferred to the VALUE array and adding to it TYPE, CLASS, and other attributes gathered from the breakpoint tables and the body of INDEXDB tape.

As this is done, it is also convenient to produce a few standard summaries which are printed when the processing of HISTAPE is complete.

At the same time all Burst/Damage events are written in formats suitable for input to the damage assessment models.

SUMMARY OF SUBROUTINES PERFORMING MAJOR FUNCTIONS

The calling hierarchy of subroutines in the program is shown in figure 61.

Subroutine READATA reads the user-input parameter cards which establish: the game start time; and the command function codes and weapon system codes used in preparing the output tapes for SIDAC and REST.

Subroutine MOREDATA retrieves from INDEXDB the breakpoint tables which hold the type and class names associated with each index number. It also retrieves other attributes such as YIELD, FFRAC, and PDUD for items in class WARHEAD, and CEP, LAT, LONG, DESIG, and TASK from other appropriate items.

Subroutine READOUT reads each event (HISTOUT array) from HISTAPE. Data which are specified for each event (SIDE, TIME, EVENT, IREG, IALERT, TYPE, and CLASS) are immediately placed in the VALUE array. One of the

Table 18. AGZ Tapes for SIDAC and WPNSUM (AGZREDSW and AGZBLUSW) (Sheet 1 of 2)

COLUMN	INFORMATION	REMARKS
1	S	Constant
2	Ø	Constant
3	Command or function code	1-9
4-8	QUICK index number (INDEXNO)	00001-12000
9-10	Day	01-31
11-12	Hour	00-23
13-14	Minutes	00-59
15-16	Month	01-12
17-18	Year	00-99
19-24	Latitude	6 Numeric (Deg., Min., Sec.)
25	North or South	N or S
26-32	Longitude	7 Numeric (Deg., Min., Sec.)
33	East or West	E or W
34-38	Target designator	2 Alpha 3 Numeric
39-44	Blank	
45	Region code	1-3
46-48	Fission/yield ratio	000-1000
49	Blank	
50-54	Yield (KT)	00001-99999
55	HOB	A or G
56-57	Blank	
58-60	CEP	All are 000
61	Blank	
62-63	Task/subtask	2 Alpha
64-65	Country location	2 Alpha
66-69	Blank	
70-71	Warhead (code)	00-99

Table 18. (cont.)
(Sheet 2 of 2)

COLUMN	INFORMATION	REMARKS
72-73	Vehicle type (code)	00-99
74-79	Unit sortie	6 Numeric
80	Blank	

Table 19. AGZ Tapes for REST and OUTBLUE (AGZREDRO and AGZBLURO) (Sheet 1 of 2)

COLUMN	INFORMATION	REMARKS
1	S	Constant
2	Ø	Constant
3	Command or function code	1-9
4-8	QUICK index number (INDEXNO)	00001-12000
9-10	Day	01-31
11-12	Hour	00-23
13-14	Minutes	00-59
15-16	Month	01-12
17-18	Year	00-99
19-24	Latitude	6 Numeric (Deg., Min., Sec.)
25	North or South	N or S
26-32	Longitude	7 Numeric (Deg., Min., Sec.)
33	East or West	E or W
34-38	Target designator	2 Alpha 3 Numeric
39-44	Blank	
45	Region code	1-3
*46-48	Fission/yield ratio	000-100
49	Blank	
*50-54	Yield (KT)	00001-99999
55	HOB	A or G
56-57	Blank	
58-60	CEP	All are 000
61	Blank	
62-63	Task/subtask	2 Alpha

* When yield (CC 50-54) is less than 100, the field for fission/yield ratio (CC 46-48) must be blank.

Table 19. (cont.)
(Sheet 2 of 2)

COLUMN	INFORMATION	REMARKS
64-65	Country location	2 Alpha
66-68	Blank	
*69-71	Weapon system	3 Alpha
72-73	Vehicle type (code)	01-99
74-79	Unit sortie	6 Numeric
80	Blank	

*Weapon system (CC 69-71) is derived from command/function code (CC 3). (See Weapon System Code Cards, User-Input Parameters, Program READSUM, Chapter 5, User's Manual, Volume II.)

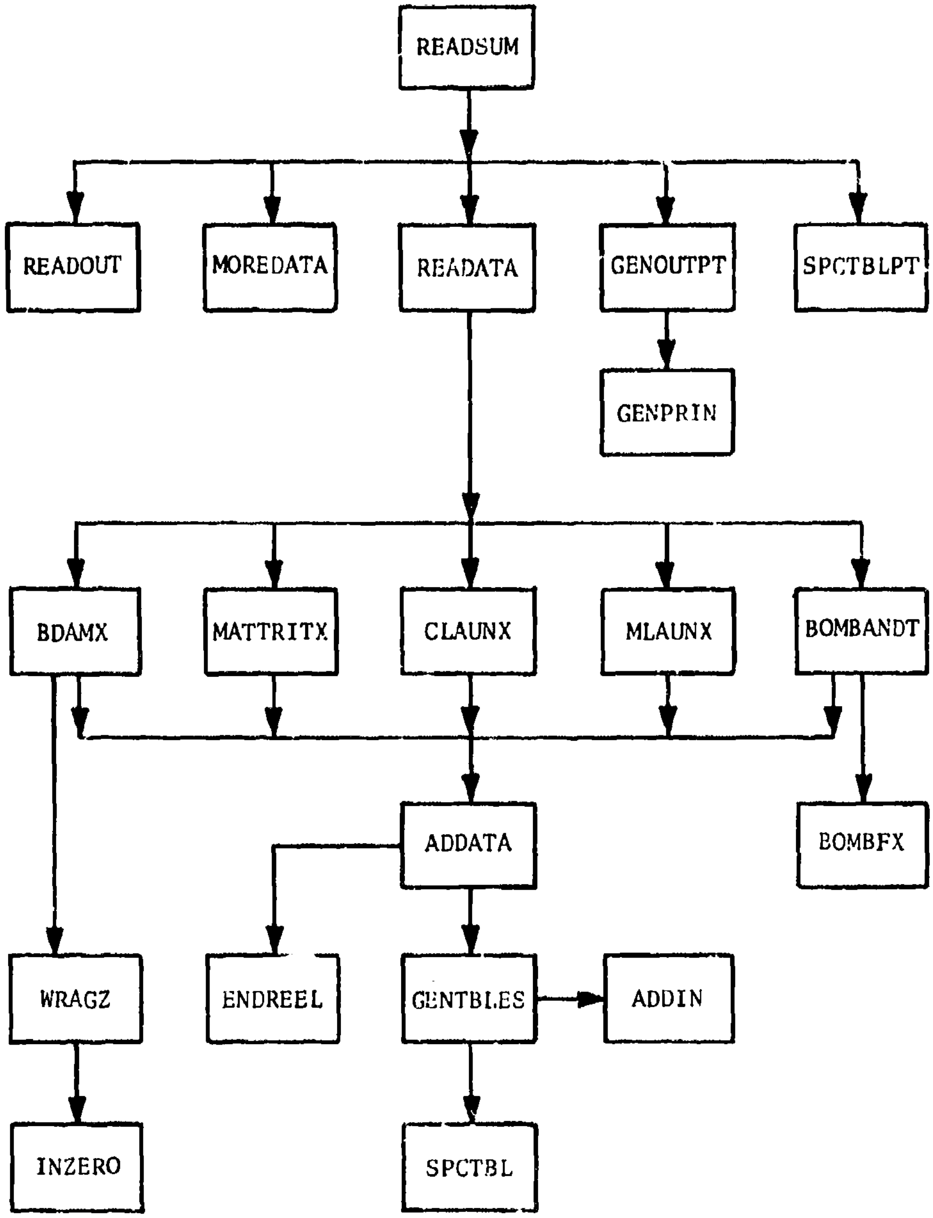


Fig. 61. Calling Hierarchy of Program READSUM

following subroutines is then called to transfer information which pertains only to that event.

Subroutine MLAUNX is called for event 1. Attributes are added to the VALUE array depending on whether or not a command failure has occurred. If it has not, each separate missile launch is examined and attributes describing it are filled.

Subroutine MATTRITX is called for events 9 and 18 to transfer information to VALUE for Area and Terminal Attrition events.

Subroutine CLAUNX is called for event 3 to add attributes for a Complete Launch event to the VALUE array.

Subroutine BOMBANDT is called for events 2, 4, 5, 7, 8, 11-17, and 19-21 to add attributes for bomber and tanker events.

Subroutine BOMBFX is called by BOMBANDT to process events which were planned but did not occur because of bomber or tanker failure.

Subroutine BDAMX is called for event 10 to add the attributes associated with the delivery vehicle and the target in a Burst/Damage event.

Subroutine WRAGZ is called by BDAMX to write the Burst/Damage event data in formats suitable for input to the various damage assessment models.

Subroutine ADDATA is called by BDAMX, MATTRITX, MLAUNX, CLAUNX, and BOMBANDT to add the information retrieved from INDEXDB by MOREDATA to the VALUE array, to write HISTDATA and HISTAGZ, and to call the subroutines which accumulate the data for the standard summaries.

Subroutine GENTBLES is called by ADDATA to update the summary tables of bomber, tanker, and missile data.

Subroutine SPCTBL is called by GENTBLES to store data for the YIELD and AGZ summaries.

Subroutine GENOUTPT combines data accumulated for the bomber, tanker, and missile summaries with labeling information and calls subroutine GENPRIN to print these tables.

Subroutine SPCTBLPT prints the cumulative megaton and cumulative AGZ tables.

Subroutine GENPRIN prints the bomber, tanker, and missile standard summaries.

COMMON BLOCK DEFINITION

External Common Blocks

The external common blocks used by program READSUM in processing INDEXDB and HISTAPE are shown in table 20. Program READSUM also references the following utility routine common blocks which are described in appendix A of Volume I, Part A, Programming Specifications Manual, Data Input Subsystem: /ITP/, /TAPES/, /NOPRINT/, /MYIDENT/, /FILABEL/, /PROCESS/, /EDITERM/, /EDITAPE/, and /TWORD/. The filehandler common block /FILE/ described on page 17 of the same volume is also used to determine the number of words written on HISTDATA and HISTAGZ.

Internal Common Blocks

The internal common blocks used by program READSUM are described in table 21.

Table 20. Program READSUM External Common Blocks
(Sheet 1 of 2)

INPUT FROM INDEXDB AND HISTAPE

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
2	KNTRY(256)	List of distinct CNTRYOWN, CNTRYLOCs
	KFCN(256)	List of distinct FUNCTIONs
	KREG(11)	List of distinct IREGs
	KLATPK(8000)	Packed latitude and longitude
	LATLONG(1)	Intermediate packing word
3	NDESTK(1)	Intermediate DESIG, TASK packing word
ASM	CEPA(20)	ASM CEP by type
	IWHDA(20)	ASM WHDTYPE by type
BYTYPE	CUMNO(15)	Cumulative number of types
	BTYPES(15)	Smallest index number in each class
	INDCLAS(15)	Beginning index number for each class
	INDBEG(250)	Beginning index number for each type
	TYPENAME(250)	Array containing type names
	KLASSES(250)	Array containing class name for each type
BOMBER	CEPB(40)	Bomber CEP by JTYPE
MISSILE	CEPM(40)	Missile CEP by JTYPE

* Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 20. (cont.)
(Sheet 2 of 2)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
WARHEAD	YIELDX(50)	Yield by WHDTYPE
	FFRACX(50)	Fission/fraction by WHDTYPE
	PDUDX(50)	Probability of dud warhead
HISTOUT	NHISTOUT(200)	Block of data describing each event simulated by SIMULATE; see chapter 2 of this volume, The Simulation Subsystem, Output Files

Table 21. Program READSUM Internal Common Blocks
(Sheet 1 of 4)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
1	NDXCTY(12000)	For each INDEXNO it contains packed indices to arrays in common block /2/
BCDKLS	BCDBOM	Class name BOMBER
	BCDMSL	Class name MISSILE
	BCDTNK	Class name TANKER
	BCDASM	Class name ASM
	BCDWHD	Class name WARHEAD
COMCOD	COMCOD(160)	Command codes for each type
DOAGZ	NDOAGZ	Input specification to control producing strike tape
	NPRAGZ	Input specification to control printing of strike tape
	NDNSW	Switch to control writing of strike tapes
	NPSW	Switch to control printing of strike tapes
IREGS	IREGT	Target region
KEYS	KLAT	LATITUDE packing-unpacking key
	KLONG	LONGITUDE packing-unpacking key
	KEYOWN	CNTRYOWN packing-unpacking key
	KEYLOC	CNTRYLOC packing-unpacking key
	KEYFCN	FUNCTION packing-unpacking key

*Parenthetical values indicate array dimensions. All other elements are single word variables.

Table 21. (cont.)
(Sheet 2 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
KEYS (cont.)	KEYLAT	LAT-LONG index packing-unpacking key
	KEYREG	IREG packing-unpacking key
KNTWRT	NOWRTD	Number of words on HISTDATA
	NOWRTA	Number of words on HISTAGZ
MAXES	MAXKLS	Maximum number of classes
	MAXWHD	Maximum number of warheads
	MAXASM	Maximum number of ASMs
	MAXMIS	Maximum number of missiles
	MAXBOM	Maximum number of bombers
	MAXLOC	Maximum INDEXNO
	MAXREG	Maximum number of regions
	MAXGRP	Maximum number of country groups
OUTPUTSM	VC1(16,2)	BOMBER type names by side
	VC2(16,2)	MISSILE type names by side
	VC3(2,2)	TANKER type names by side
	NCT1(2)	Number of type names in VC1
	NCT2(2)	Number of type names in VC2
	NCT3(2)	Number of type names in VC3
	A(27,16,2)	BOMBER data array
	AM(14,16,2)	MISSILE data array
	AT(14,2,2)	TANKER data array
	NODO1(2)	Switches to indicate overflow of VC1
	NODO2(2)	Switches to indicate overflow of VC2

Tablo 21. (cont.)
(Sheet 3 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
OUTPUTSM (cont.)	NODO3(2)	Switches to indicate overflow of VC3
SPECIAL	NAGZ(12,10,2)	Cumulative number of AGZs by region
	YDEL(12,10,2)	Cumulative yield delivered by region
	TAU(10)	Time intervals for summaries
SPECIAL (cont.)	NAMESIDE(2)	SIDE labels
	NAMEREG(12,2)	Region labels
	NUMREG	Number of regions
	MAXTIM	Maximum number of time intervals
	NAGZ2(12,10,2)	Cumulative number of AGZs by country group
	YDEL2(12,10,2)	Cumulative yield delivered by country group
	NUMGP(10,2)	Number of countries in each country group
	CCODES(20,10,2)	Names of countries in each country group
	NAMEGP(10,2)	Country group labels
ST	KFAIL	Contains default value
	NAMESIDE(2)	Contains BLUE, RED
STIME	JDAY	Game starting day
	JMON	Game starting month
	JYEAR	Game starting year
	JHOUR	Game starting hour
	JMIN	Game starting minute

Table 21. (cont.)
(Sheet 4 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
TAPES	ITAPES(10)	Logical tape numbers
	MAXTAPE(10)	Presently unused
	NTP	Logical tape number
	NROUT	Number of output tapes
TAPETYPE	NTPTYPE	Indicates type of HISTAPE
TBLSIZE	MAXROWB	Maximum rows for bomber tables
	MAXROWM	Maximum rows for missile tables
	MAXROWT	Maximum rows for tanker tables
	MAXCOLB	Maximum columns for bomber tables
	MAXCOLM	Maximum columns for missile tables
	MAXCOLT	Maximum columns for tanker tables
TITLES	TITLES(81)	Row labels for bomber tables
	TITLEM(42)	Row labels for missile tables
	TITLET(42)	Row labels for tanker tables
WPSYS	NWPSYS(9,2)	Weapon system codes by function code and side

PROGRAM READSUM

PURPOSE: To initialize tapes and call various subroutines required to produce HISTDATA and HISTAGZ, the standard summaries, and the four strike tapes.

ENTRY POINTS: READSUM

FORMAL PARAMETERS: None

COMMON BLOCKS: FILABEL, ITP, KNTWRT, MYIDENT, NOPRINT, STIME, TAPES, TAPETYPE

SUBROUTINES CALLED: ENDDATA, GENOUTPT, INITAPE, MOREDATA, NEWUNIT, READATA, READOUT, SETREAD*, SPCTBLPT, STORAGE, WRAGZF

CALLED BY: None

Method

Program READSUM calls subroutine MOREDATA to read the breakpoint tables and other required data from INDEXDB. Subroutine READOUT reads HISTAPE and controls the calling of the event processing routines. Subroutines GENOUTPT and SPCTBLPT are called to print the standard summaries. Subroutine WRAGZF is called to complete the writing of the four strike tapes.

Program READSUM is illustrated in figure 62.

---

*See filehandler routines

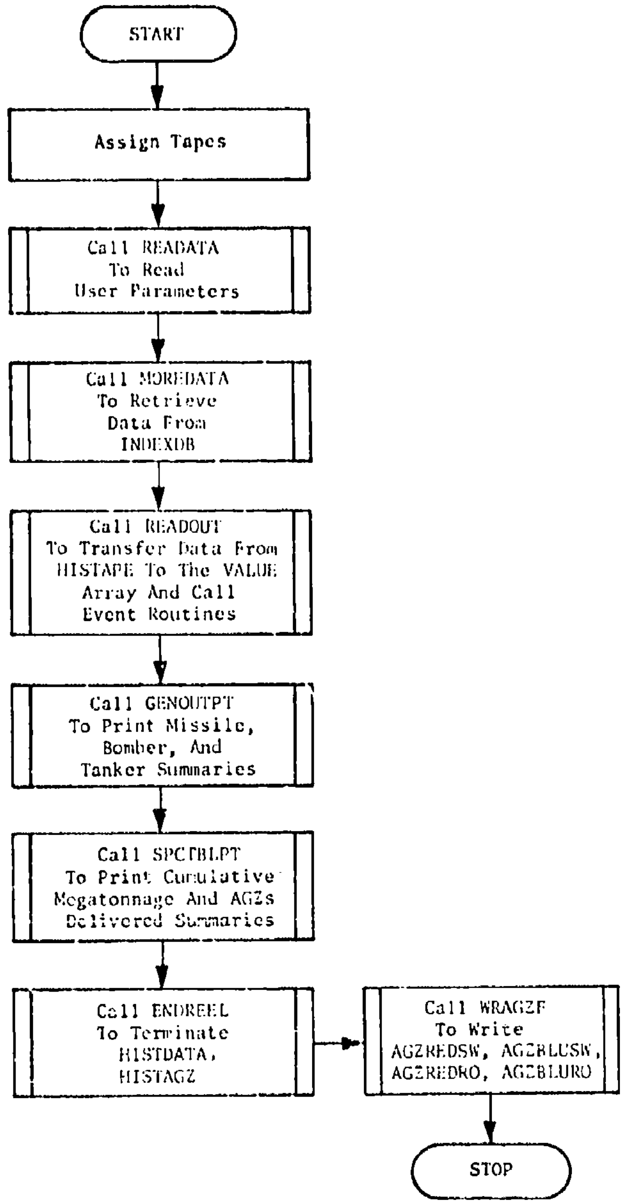


Fig. 62. Program READSUM

SUBROUTINE ADDATA

PURPOSE:	To add the information retrieved by MOREDATA to the VALUE array and to control the writing of the HISTDATA and HISTAGZ tapes.
ENTRY POINTS:	ADDATA, ADDREP, RESTORE
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BYTYPE, EDITERM, EDITAPE, FILE*, IREGS, KEYS, KNTWRT, 1, 2, 3, MAXES, MYIDENT, PROCESS, TAPES, WARHEAD
SUBROUTINES CALLED:	CHANGE, ENDREEL, GENTBLES, IGET, OUTITEM, PRITEM
CALLED BY:	BDAMX, BOMBANDT, CLAUNX, MATTRITX, MLAUNX

Method

Subroutine ADDATA is called by each of the event translator subroutines. If there are nonvarying attributes defined for a sequence of events (as, for example, Burst/Damage events for a collocation island), entry is through ADDREP, which sets a switch to prevent default values from being restored before exit from the subroutine. Entry point RESTORE is provided to reset the default values after the events with nonvarying attributes have been processed.

ADDATA begins by calling function IGET to unpack the indices of the attributes FUNCTION, CNTRYOWN, CNTRYLOC, and IREG, corresponding to the current value of INDEXNO (statement 101). These attributes are then obtained and placed in the VALUE array. If a target index (INTAR) has been assigned, the appropriate indices are obtained by IGET to retrieve the attributes CNTYOWNT and CNTYLOCT, which are then stored in VALUE (statement 111). IREGT is stored in a common block /IREGS/ for transmittal to the necessary subroutines. The breakpoint tables are used to determine the class and type corresponding to INTAR. These attributes also are added to array VALUE. If the index WHDTYPE has been defined, the attributes PDUD and YIELD are retrieved from the arrays in which they were stored by MOREDATA (statement 131); if the index WHDTYPEN is

*See filehandler routines.

defined instead, only the attribute YIELD is retrieved (statement 141). The appropriate attributes then are placed in VALUE.

GENTBLES is called to store data for the standard summaries (statement 150). All data in the VALUE array now are written onto the HISTDATA tape. If the event being processed involved a weapon burst (which was indicated by setting PRIMETAR to 1), the data also are written onto the HISTAGZ tape (statement 151). Default values are reset if entry was through ADDATA (statement 152). In order to provide for multiple reel data base tapes, the number of words written on the HISTDATA tape thus far is queried. When this number exceeds one million, subroutine ENDREEL is called to have the operator mount a new reel of tape. Otherwise, control is returned to the calling subroutine.

Subroutine ADDATA is illustrated in figure 63.

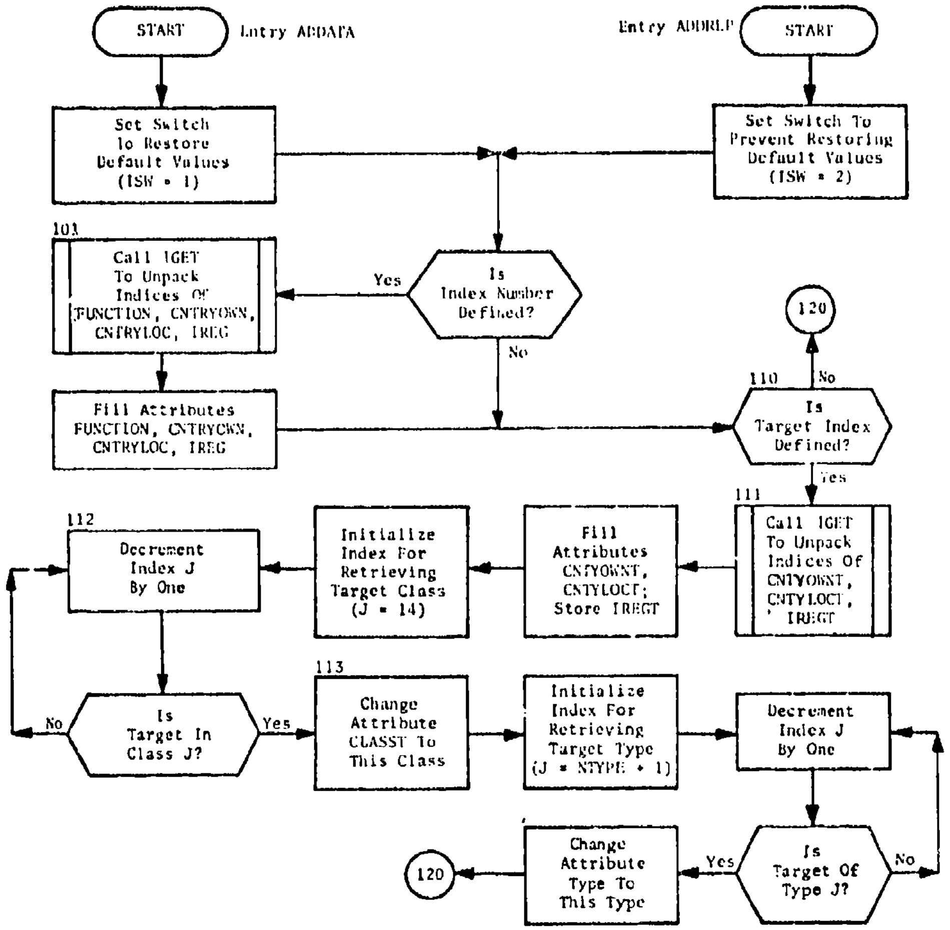


Fig. 63. Subroutine ADDATA (Entry ADDREP)
(Sheet 1 of 2)

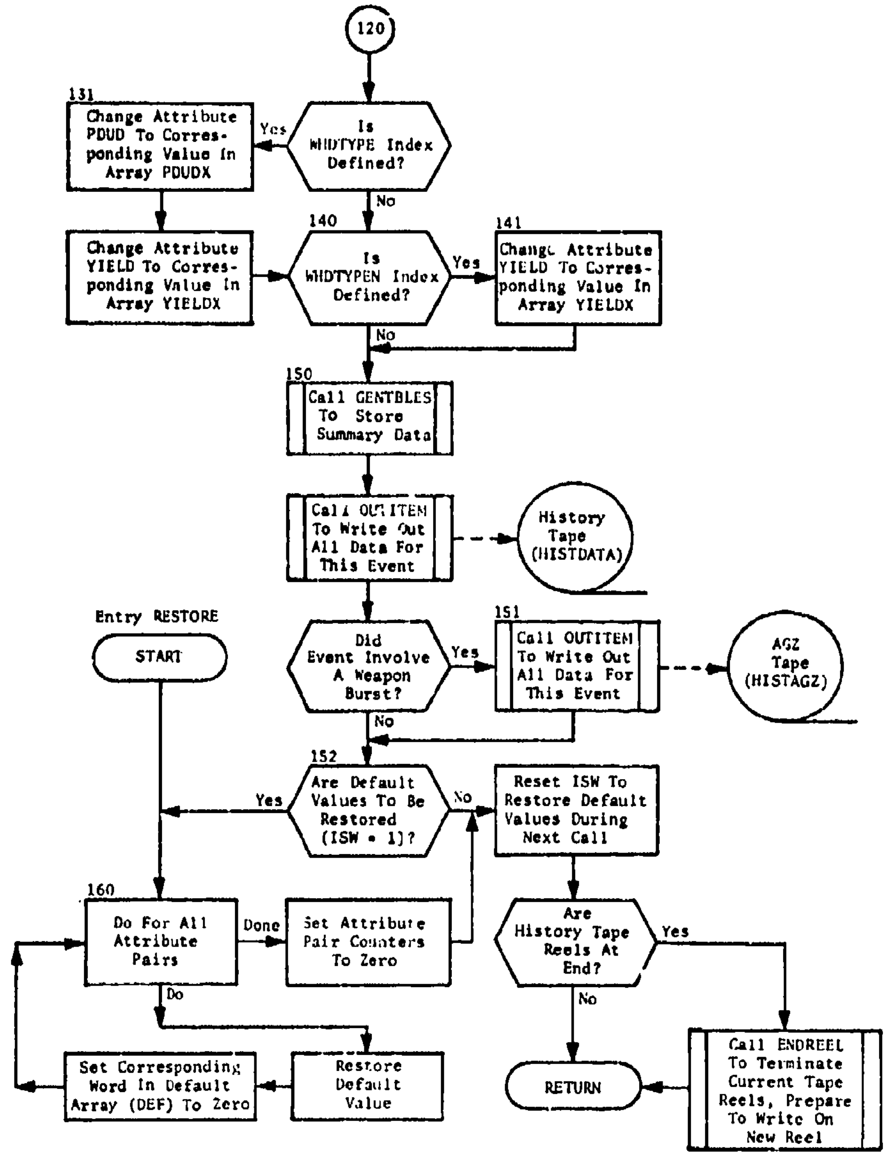


Fig. 63. (cont.)
(Sheet 2 of 2)

SUBROUTINE ADDIN

PURPOSE:	To increment the appropriate entries in the bomber, tanker, and missile standard summaries.
ENTRY POINTS:	ADDIN
FORMAL PARAMETERS:	V - Type name array AR - Array of tabled summary data NCT - Number of entries in V NODO - Switch to indicate overflow of array V K - Index to side I - Row index of array AR M - Number of rows in array AR N - Number of columns in array AR
COMMON BLOCKS:	EDITAPE, EDTTERM, PROCESS
SUBROUTINES CALLED:	None
CALLED BY:	GENTBLES

Method

Subroutine ADDIN examines the table of type names to determine whether the current type is present (statement 73). If it is, the index becomes the column index of the table and the table entry is incremented (statement 72). If it is not, the new type is added to the array of type names if there is space available (statement 777), and this index becomes the column index.

The appropriate entry in the output array (statement 773) is incremented and control returns to GENTBLES.

Subroutine ADDIN is illustrated in figure 64.

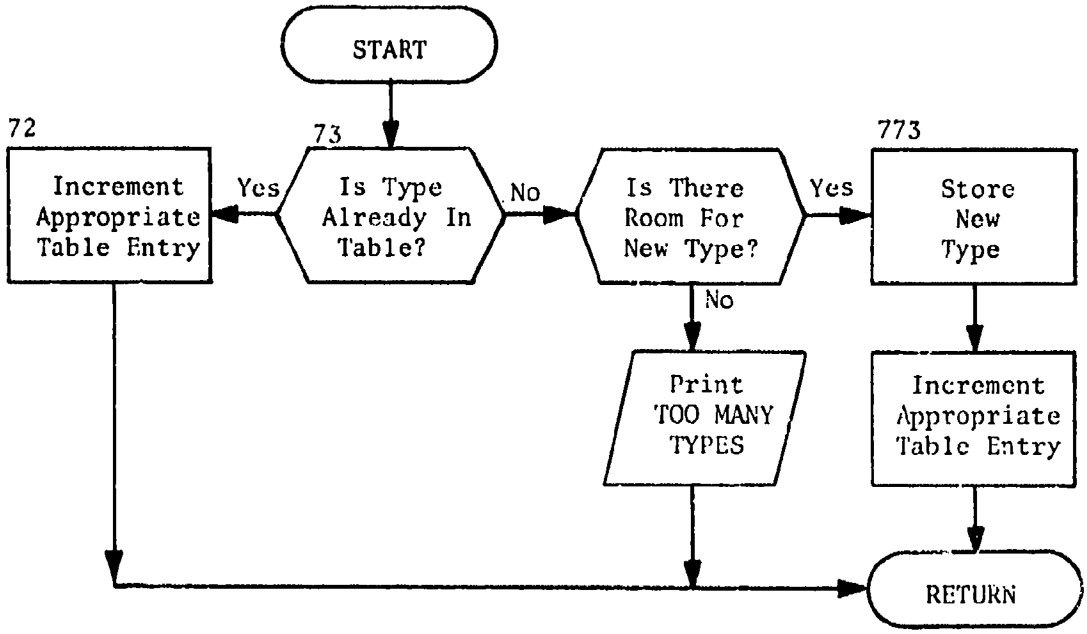


Fig. 64. Subroutine ADDIN

PURPOSE: To add the attributes associated with the delivery vehicle and the target in a Burst/Damage event to the VALUE array.

ENTRY POINTS: BDAMX

FORMAL PARAMETERS: None

COMMON BLOCKS: EDITAPE, EDITERM, HISTOUT, MAXES, PROCESS, TAPETYPE, TWORD

SUBROUTINES CALLED: ADDREP, CHANGE, IGET, KEYMAKE, RESTORE, WRAGZ

CALLED BY: READOUT

Method

BDAMX begins by adding the index number, vehicle index number, and the warhead type (INDEXNO, INDV, and WHDTYPE) to the VALUE array. Function IGET then is called to unpack the target index (INTAR) and the outcome code (CODE, see table 22), which were packed by the Simulator for each target. These attributes also are added to the VALUE array. If the target is the first member of a collocation island or an individual target, PRIMETAR is set to one (statement 4), ground zero data (DGX, DGY, DHOB, AGX, AGY, and AHOB) are added to the VALUE array, and subroutine WRAGZ is called. PRIMETAR is set to zero for the remaining members of a collocation island. For all targets, subroutine ADDREP is called to complete the transfer of data without restoring default values (statement 8). After all targets in the collocation island have been processed, a call is made on RESTORE to reset these default values and control is return to READOUT.

Subroutine BDAMX is illustrated in figure 65.

Table 22. Outcome Code (Attribute: CODE) for BDAMX

CODE	DESCRIPTION
1	Target survives
2	No assessment necessary
3	Target killed
4	Target already dead

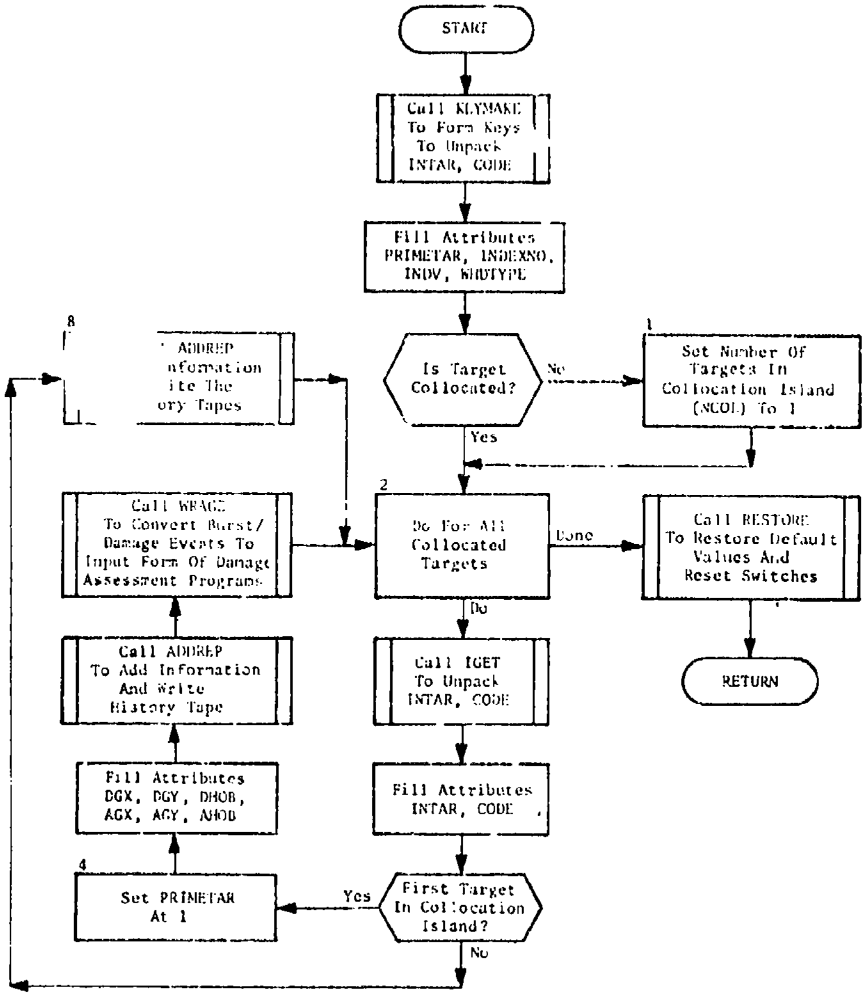


Fig. 65. Subroutine BDAMX

SUBROUTINE BOMBANDT

PURPOSE: To add the attributes for bomber and tanker events to the VALUE array.

ENTRY POINTS: BOMBANDT

FORMAL PARAMETERS: None

COMMON BLOCKS: BCOKLS, EDITAPE, EDITERM, HISTOUT, PROCESS, TAPETYPE

SUBROUTINES CALLED: ADDATA, BOMBFX, CHANGE

CALLED BY: READOUT

Method

BOMBANDT begins by adding to the VALUE array the attributes which apply to all bomber and tanker events. These attributes include the index number, vehicle index number, zone index number, number of decoys carried by the vehicle, altitude index, number of countermeasures carried by the vehicle, delay time, and outcome code (see table 23), (INDEXNO, INDV, ZONE, NDECOYS, IALT, NCM, DELAY, and BCODE). For Local Attrition events, a flag (IDUD) which indicates whether or not the warhead was a dud also is added to VALUE (statement 8); for ASM Launch events, the ASM type index (ASMTYPE) is filled into the array. If the event is not a bomber or tanker Launch or Local Attrition event, the geographic location (PLACE) is included in the list of attributes added (statement 9). Subroutine ADDATA then is called if the vehicle survived the event (statement 1); otherwise a call is made on subroutine BOMBFX (statement 2). Upon return from either subroutine, control reverts to READOUT.

Subroutine BOMBANDT is illustrated in figure 66.

Table 23. Bomber and Tanker Outcome Codes
(Sheet 1 of 2)

NBCODE	DESCRIPTION
1	Success
2	Launch base dead
3	Takeoff abort
4	Refuel abort
5	No tankers available
6	Penetrate enemy territory
7	Leave enemy territory
8	Killed by area attrition
9	Killed by local attrition
10	Killed by local attrition after warhead release
11	ASM killed by local attrition
12	Tanker abort in refuel area
13	Departure of full tanker
14	Random abort
15	ASM launch abort
16	Not used
17	Recovery base dead on arrival
18	Scheduled splash
19	Abort on first of two refuelings
20	No tankers on first of two refuelings
21	No live recovery base at depenetration time
22	Recovery base saturated
23	Killed after recovery
24	Killed after recovery at home base
25	Recovery at home base
26	Home recovery base dead on arrival
27	Arrival of ASM at target

Table 23. (cont.)
(Sheet 2 of 2)

NBCODE	DESCRIPTION
28	Successful first of two refuelings
29	Refuel abort, alternate mission return home
30	No tankers available, alternate mission return home

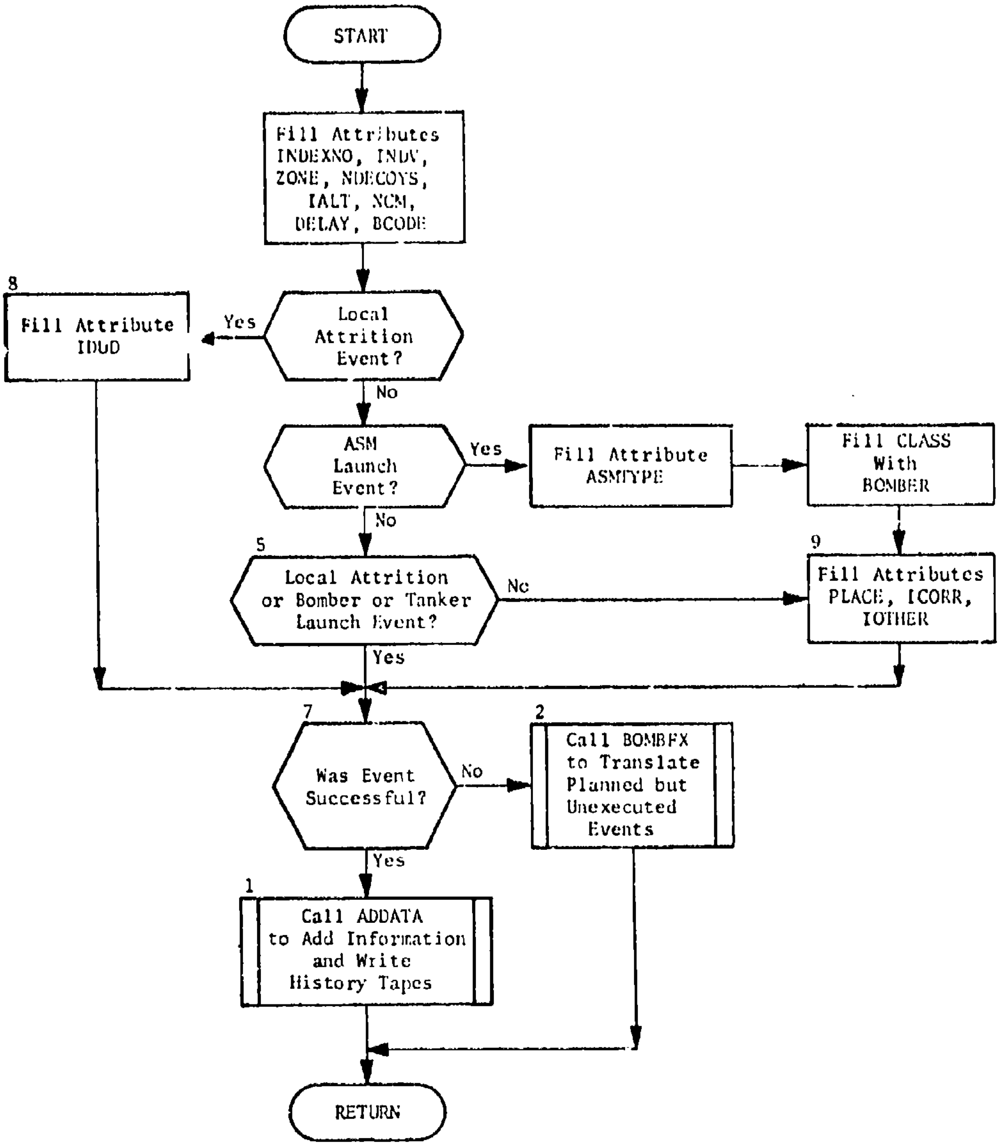


Fig. 66. Subroutine BOMBANDT

SUBROUTINE BOMBFX

PURPOSE: To translate events which were planned but did not occur because of bomber or tanker failure.

ENTRY POINTS: BOMBFX

FORMAL PARAMETERS: None

COMMON BLOCKS: EDITAPE, EDITERM, HISTOUT, PROCESS

SUBROUTINES CALLED: ADDREP, CHANGE, RESTORE

CALLED BY: BOMBANDT

Method

BOMBFX begins by calling subroutine ADDREP to output the information already obtained by BOMBANDT for the event before it failed. A loop then is entered to add the attributes associated with events which had been planned but never took place (statement 10). For all bomber and tanker events, except ASM Launches, the time increment, place index, and event number remain in the History Table which was generated by the Simulator (under TINCN, INDPN, and INDEN, respectively (statement 13)). In addition, a warhead type index which is required for Local Attrition events is stored in a separate Warhead Table (NWTYPEN)(statement 5). As each event except an ASM Launch is encountered, the above information is retrieved and added to the VALUE array (TIMEN, EVENTN, WHDTYPEN, and PLACEN). The item then is output by calling subroutine ADDREP (statement 11).

For an ASM Launch event, the ASM type index (ASMTYPE) is stored in the History Table location otherwise reserved for the place index (INDPN). In addition, the Launch event always is followed by a Local Attrition event for which the event number has not been specified. Therefore, whenever an ASM Launch event is encountered, the attribute ASMTYPE is changed to the index stored under INDPN (statement 12), and the Local Attrition event number (8) is placed in the next location of INDEN. The processing continues as above until all items in the History Table have been considered. Subroutine RESTORE then is called to reset the default values and control is returned to BOMBANDT.

Subroutine BOMBFX is illustrated in figure 67.

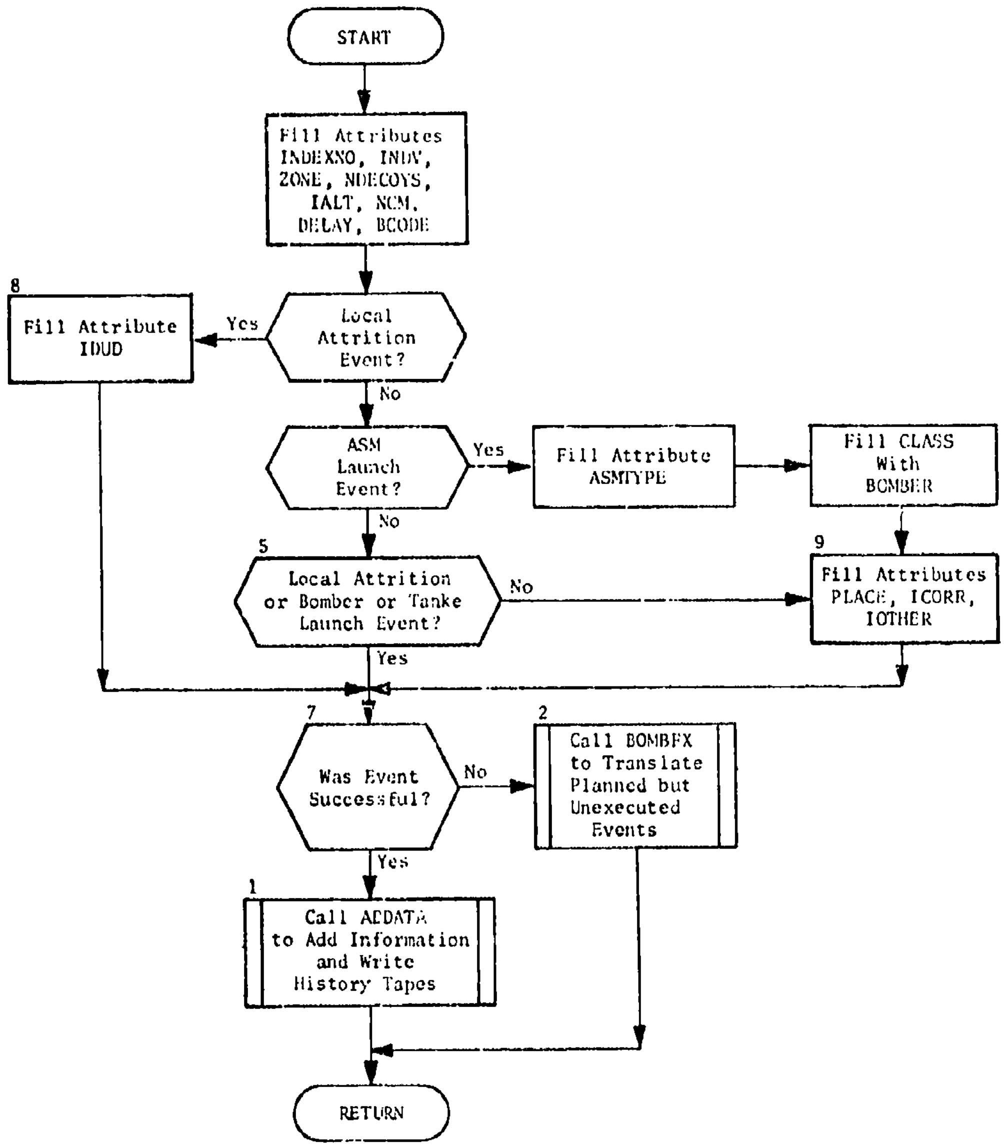


Fig. 66. Subroutine BOMBANDT

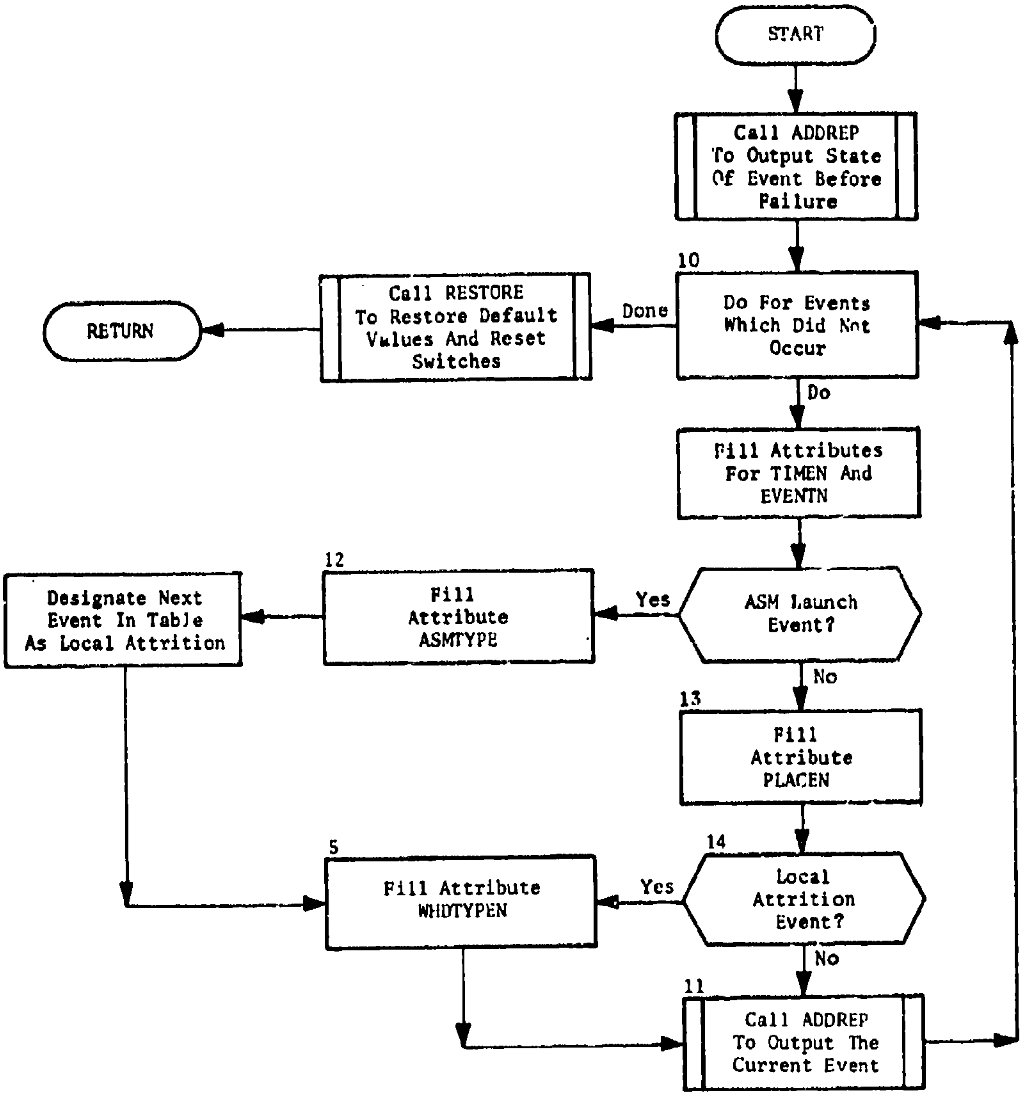


Fig. 67. Subroutine BOMBFX

SUBROUTINE CLAUNX

PURPOSE: To add attributes for a Complete Launch event to the VALUE array.

ENTRY POINTS: CLAUNX

FORMAL PARAMETERS: None

COMMON BLOCKS: EDITERM, EDITAPE, HISTOUT, PROCESS

SUBROUTINES CALLED: ADDATA, CHANGE

CALLED BY: READOUT

Method

CLAUNX adds the attributes which represent index number, vehicle index number, target index number, warhead type index, and the outcome code (INDEXNO, INDV, INTAR, WHDTYPE, and CODE) to the VALUE array for each Complete Launch event. Subroutine ADDATA then is called to complete the transfer of information, and control is returned to READOUT.

Subroutine CLAUNX is illustrated in figure 68.

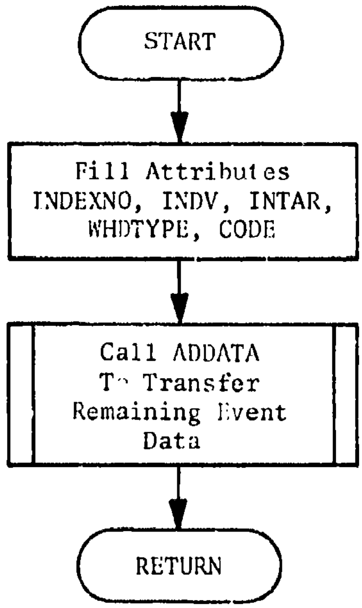


Fig. 68. Subroutine CLAUNX

SUBROUTINE ENDREEL

PURPOSE:	To enable the HISTDATA to be continued on a second reel if the first reel cannot contain the required data.
ENTRY POINTS:	ENDREEL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDITAPE, EDITERM, ITP, PROCESS, TAPES
SUBROUTINES CALLED:	CHANGE, ENDDATA, OUTITEM, SETWRITE
CALLED BY:	ADDATA

Method

Whenever more than one million words have been written on the HISTDATA currently being output, ENDREEL is called to terminate that tape and to output a message to the operator to mount a new reel. Control is returned to ADDATA.

Subroutine ENDREEL is illustrated in figure 69.

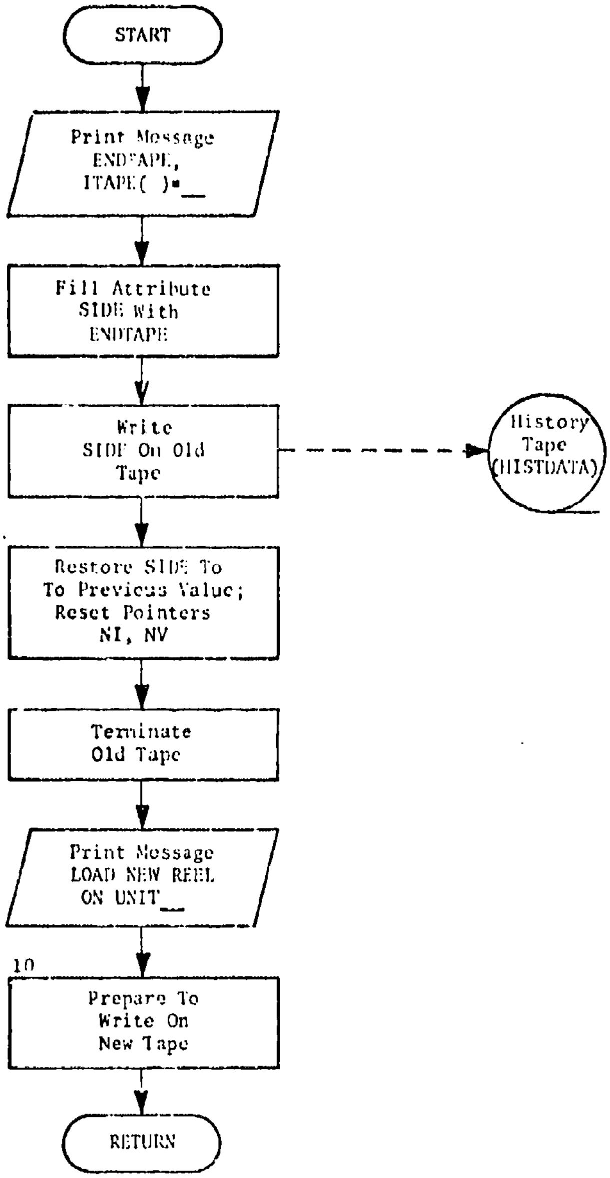


Fig. 69. Subroutine ENDREEL

SUBROUTINE GENOUTPT

PURPOSE:	To provide summary table row labels and control the printing of the bomber, missile and tanker summary tables.
ENTRY POINTS:	GENOUTPT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BCDKLS, OUTPUTSM, TBLSIZE, TITLES
SUBROUTINES CALLED:	GENPRIN
CALLED BY:	READSUM

Method

Subroutine GENOUTPT (figure 70) contains data statements which establish the row labels printed for the bomber, missile and tanker summary data accumulated by subroutine GENTBLES. Subroutine GENPRIN is called to print the summary tables for each of these classes. The following information is printed.

1. For each bomber type

   a. Number of successful launches

   b. Number of unsuccessful launches, dead base

   c. Number of unsuccessful launches, delayed launch

   d. Number of attempted first refuelings

   e. Number of aborts on first of two scheduled refuelings

   f. Number of times tankers not available for first of two scheduled refuelings

   g. Number of attempted refuelings*

---

*Applies to last scheduled refueling; i.e., to bombers to refuel only once and to the second refueling event when bombers refuel twice.

h. Number of aborts on refueling*

i. Number of times tankers not available*

j. Number refuel abort bombers returned to home base**

k. Number of bomber aborts outside enemy territory

l. Number of bombers which entered (penetrated) enemy territory

m. Number of kills by area attrition

n. Number of kills by local attrition

o. Number of ASM kills by local attrition

p. Number of aborts in enemy territory

q. Number of bombers with scheduled splashes (aborts)

r. Number of bombers which depenetrate enemy territory

s. Number of bombers which recover

t. Number of times recovery base dead

u. Number of times no live base available at depenetration time

v. Number of times recovery base saturated

w. Number of bombers killed after recovery

x. Number of bombers returned to home base

y. Number of recoveries at home base

z. Number of times base dead at recovery time

aa. Number of bombers killed after recovery at home base

2. <u>For each missile type</u>

a. Number of attempted launches

---

*Applies to last scheduled refueling; i.e., to bombers to refuel only once and to the second refueling event when bombers refuel twice.

**Applies to last scheduled refueling. Bombers aborting the first of two scheduled refuelings are returned to home base but are not counted here.

b. Number not in commission

c. Number of dead silos

d. Number not in commission, dead silo

e. Number of nondestructive aborts

f. Number of destructive aborts

g. Number of powered flight failures

h. Number of successful launches

i. Number unused

j. Number destroyed at launch

k. Number of weapons entering area defenses

l. Number of weapons entering terminal defenses

m. Number of weapons penetrating

n. Number of weapons detonating

3. <u>For each tanker type</u>

a. Number of successful launches

b. Number of unsuccessful launches, dead base

c. Number of unsuccessful launches which were rescheduled

d. Number of aborts outside refueling area

e. Number of entries into refueling area

f. Number of aborts inside refueling area

g. Number of full tankers leaving refueling area

h. Number of empty tankers leaving refueling area

i. Number of recoveries

j. Number of times base saturated

k. Number of times recovery base dead

l. Number of times no live base available at depenetration time

m. Number of tankers killed after recovery

n. Number of tankers with scheduled splash (abort)

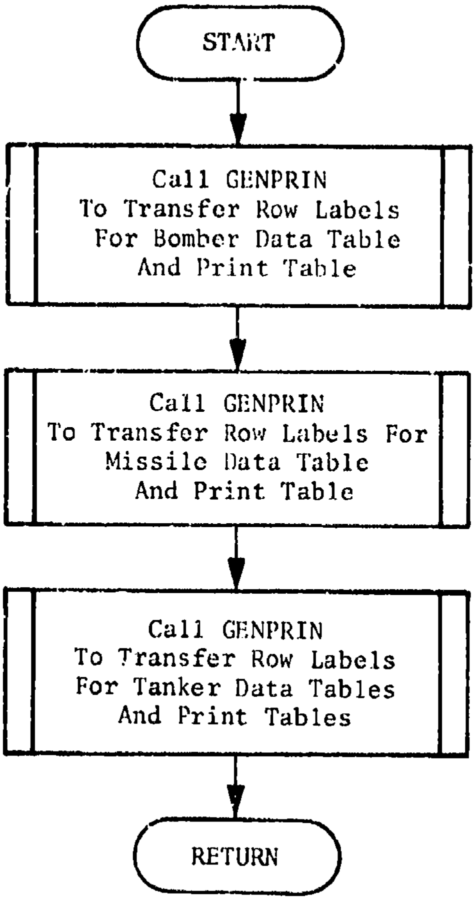


Fig. 70. Subroutine GENOUTPT

SUBROUTINE GENPRIN

PURPOSE:	To print the bomber, tanker, and missile standard summaries.
ENTRY POINTS:	GENPRIN
FORMAL PARAMETERS:	A - Array to be printed VC - Array of column labels for each side NCT - Number of entries in VC TITLE - Array containing row labels TITLE - Array containing table header label M - Number of rows N - Number of columns NA - Name of array which has variable dimensions
COMMON BLOCKS:	ST
SUBROUTINES CALLED:	None
CALLED BY:	GENOUTPT

Method

GENPRIN prints tables from data arrays accumulated by GENTBLES and labeling information supplied by GENOUTPT. If the table consists entirely of zeros, it is not printed. More than one page is printed for each table if the number of columns exceeds the maximum allowed per page (MCOL4ROW). The subroutine prints the tables for both sides and then returns control to GENOUTPT.

Subroutine GENPRIN is illustrated in figure 71.

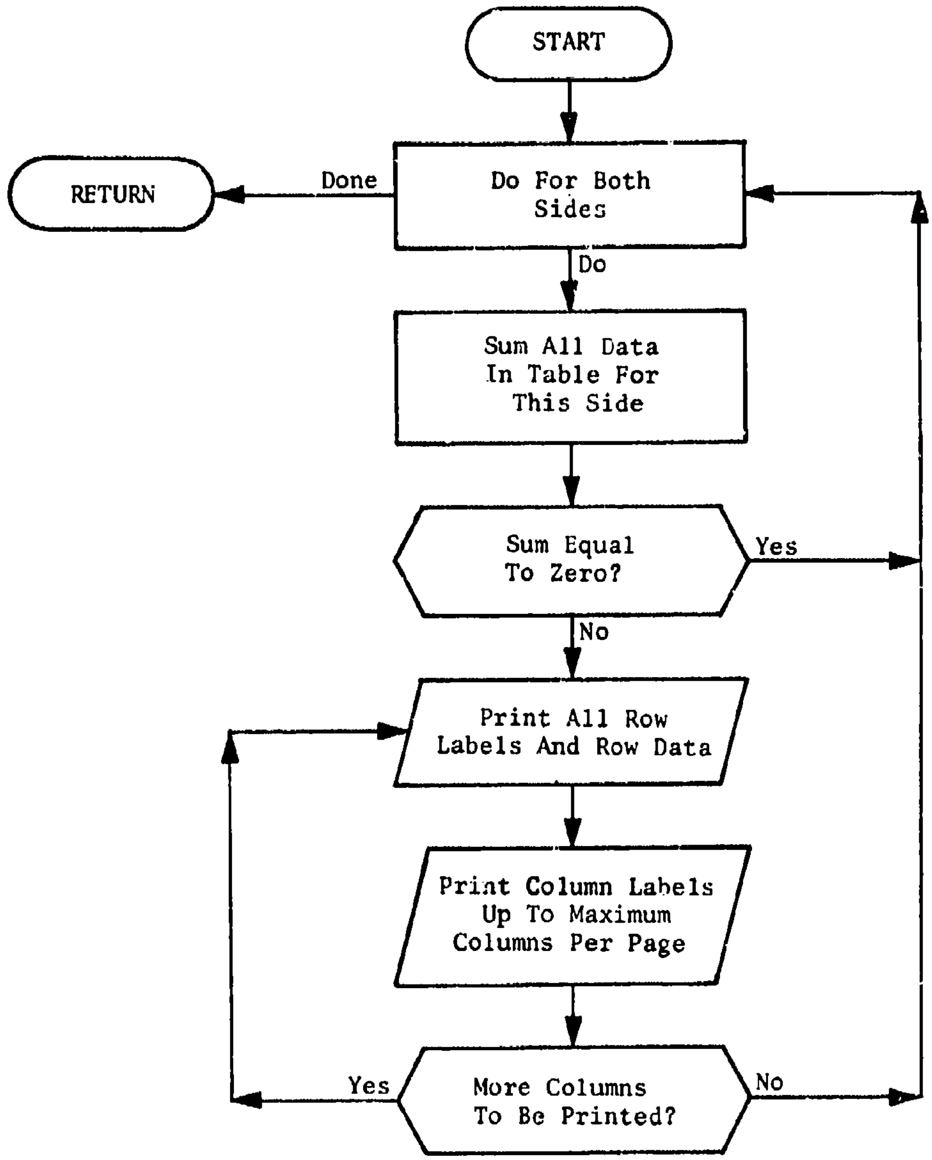


Fig. 71. Subroutine GENPRIN

SUBROUTINE GENTBLES

PURPOSE:	To examine each item which has been written on the HISTDATA tape, determine which table entries are affected, and then update these entries.
ENTRY POINTS:	GENTBLES
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BCDKLS, EDITAPE, EDITERM, OUTPUTSM, PROCESS, TBLSIZE
SUBROUTINES CALLED:	ADDIN, SPCTBL
CALLED BY:	ADDATA

Method

On the first call to GENTBLES, all arrays are cleared. Thereafter, clearing is bypassed. SPCTBL is called to store data in the other standard summaries, since GENTBLES is concerned only with the bomber, tanker, and missile summaries.

The index of the appropriate side is found (statements 90 and 91), and the indices of the entries which must be updated are determined by examining EVENTN, EVENT, and BCODE for bombers and tankers (statement 800), and MCODE for missiles (statement 301). Subroutine ADDIN is called to increment the table entries affected (statements 200, 300, and 600).

Subroutine GENTBLES is illustrated in figure 72.

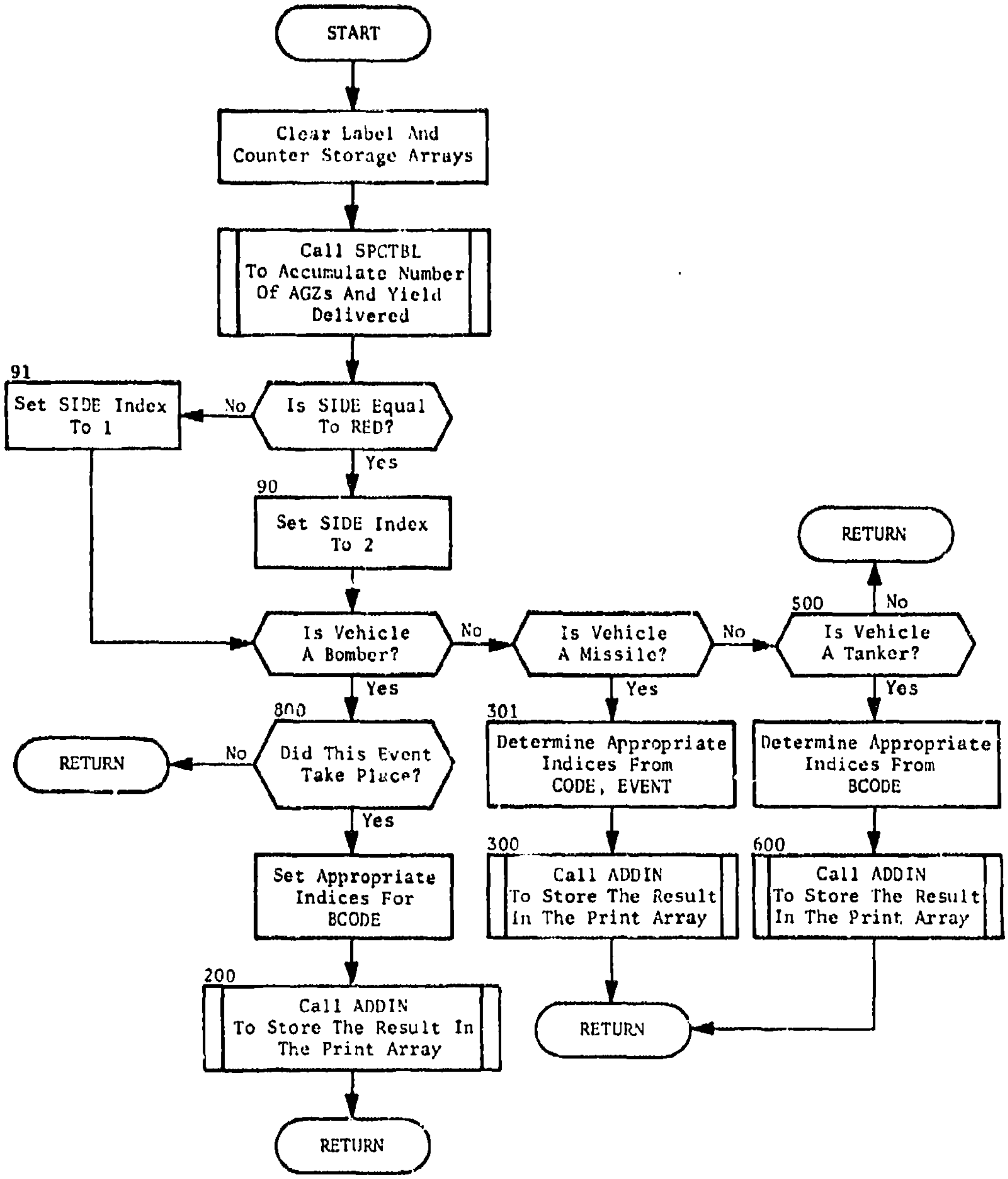


Fig. 72. Subroutine GENTBLES

SUBROUTINE INZERO

PURPOSE:	To replace each BCD blank in a word with a BCD zero.
ENTRY POINTS:	INZERO
FORMAL PARAMETERS:	M - Word in which blanks are to be replaced with zero
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	WRAGZ

Method

The eight characters of the parameter word are successively masked and compared with the octal code for a blank. When a match is found, that character is replaced by the octal code for a zero. When all eight characters have been processed, control returns to the calling program with the modified word in the A register.

The octal codes for the BCD characters blank and zero are found implicitly from a data statement.

Subroutine INZERO is illustrated in figure 73.

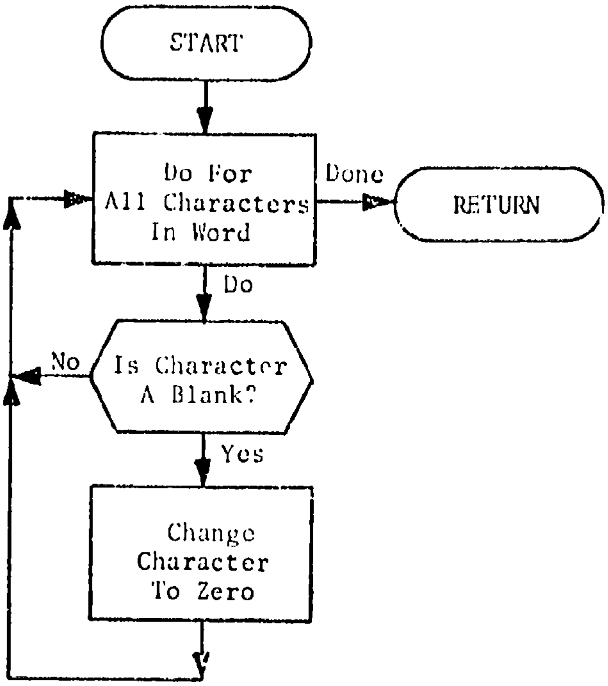


Fig. 73. Subroutine INZERO

SUBROUTINE MATTRITX

PURPOSE:	To transfer information for Area and Terminal Attrition events from the HISTOUT array to the VALUE array.
ENTRY POINTS:	MATTRITX
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDITAPE, EDITERM, HISTOUT, PROCESS
SUBROUTINES CALLED:	ADDATA, CHANGE
CALLED BY:	READOUT

Method

MATTRITX adds the index number, vehicle index number, target index number, warhead type, number of weapons, number of warheads, and number of interceptors allocated to the VALUE array (INDEXNO, INDV, INTAR, WHDTYPE, NWPNS, NWHDS, and NAL). For Area Attrition events, the number of terminal aim points, geographic location, number of surviving warheads, and outcome code (TAIM, PLACE, NWHDS, and CODE) then are filled (statement 10); for Terminal Attrition events, the number of warheads which penetrate and the number of warheads which detonate are added (NPEN and NDET) (statement 20). The transfer of data to VALUE is completed by calling subroutine ADDATA, and control is returned to READOUT.

Subroutine MATTRITX is illustrated in figure 74.

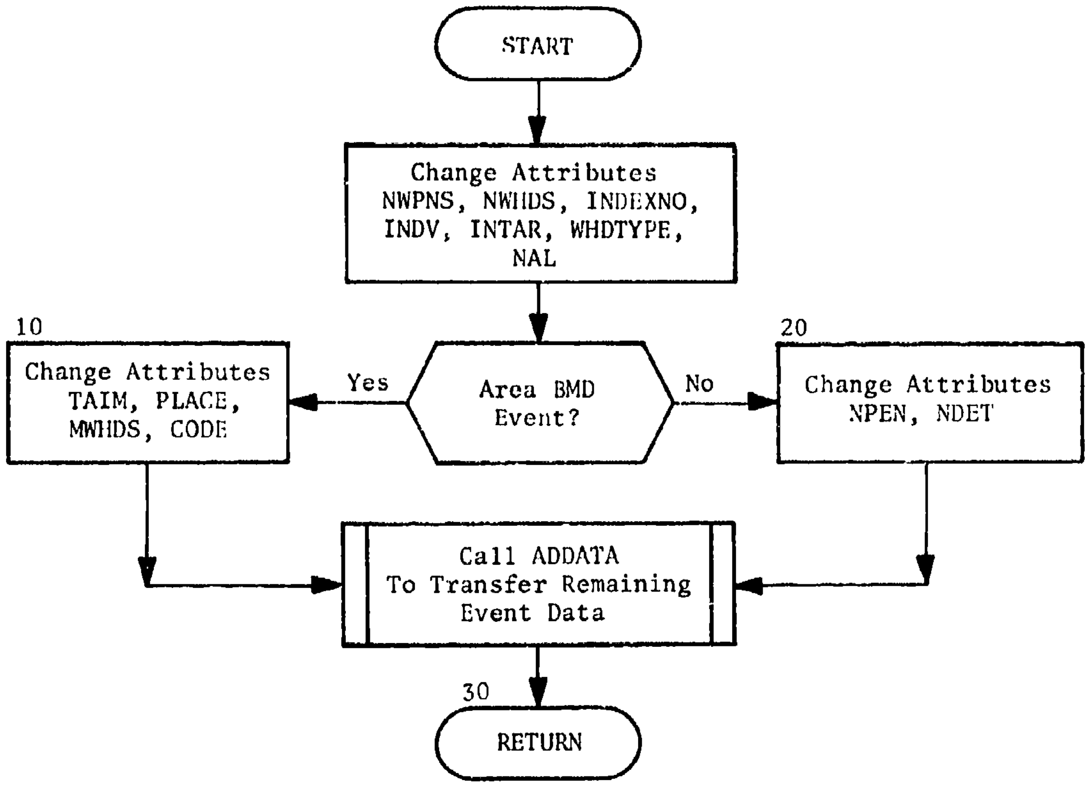


Fig. 74. Subroutine MATTRITX

SUBROUTINE MLAUNX

PURPOSE: To transfer attributes which describe missile launch events to the VALUE array.

ENTRY POINTS: MLAUNX

FORMAL PARAMETERS: None

COMMON BLOCKS: EDITAPE, EDITERM, HISTOUT, PROCESS

SUBROUTINES CALLED: ADDATA, ADDREP, CHANGE, RESTORE

CALLED BY: READOUT

Method

MLAUNX transfers to the VALUE array the warhead type (WHDTYPE) and the status of the command and control center (contained in NCODE). If NCODE indicates that a command and control failure has occurred, the total number of weapons, the index number, the vehicle index number, and the target index number (INDEXNO, INDV, and INTAR) are added to VALUE (statement 100), subroutine ADDATA is called to complete the transfer of information, and control is returned to READOUT. If command and control is in commission, the attributes representing the index number (INDEXNO), the target index number (INTAR), the vehicle index number (INDV), and the outcome code (MCODE, see table 24) are filled for each separate missile launch (statement 10). As each launch is encountered, subroutine ADDREP is called to complete the processing of that item. Then, when all launches in the event have been considered, a call is made on RESTORE to reset the default values, and control is returned to READOUT.

Subroutine MLAUNX is illustrated in figure 75.

Table 24. Outcome Codes (Attribute: MCODE) for MLAUNX

MCODE	DESCRIPTION
0	Missiles not used
1	Not in commission
2	Silo dead
3	Launch abort
4	Silo destroyed on launch abort
5	Failure in powered flight
6	Success
7	Missile planned for later launch event
8	Not in commission and silo dead

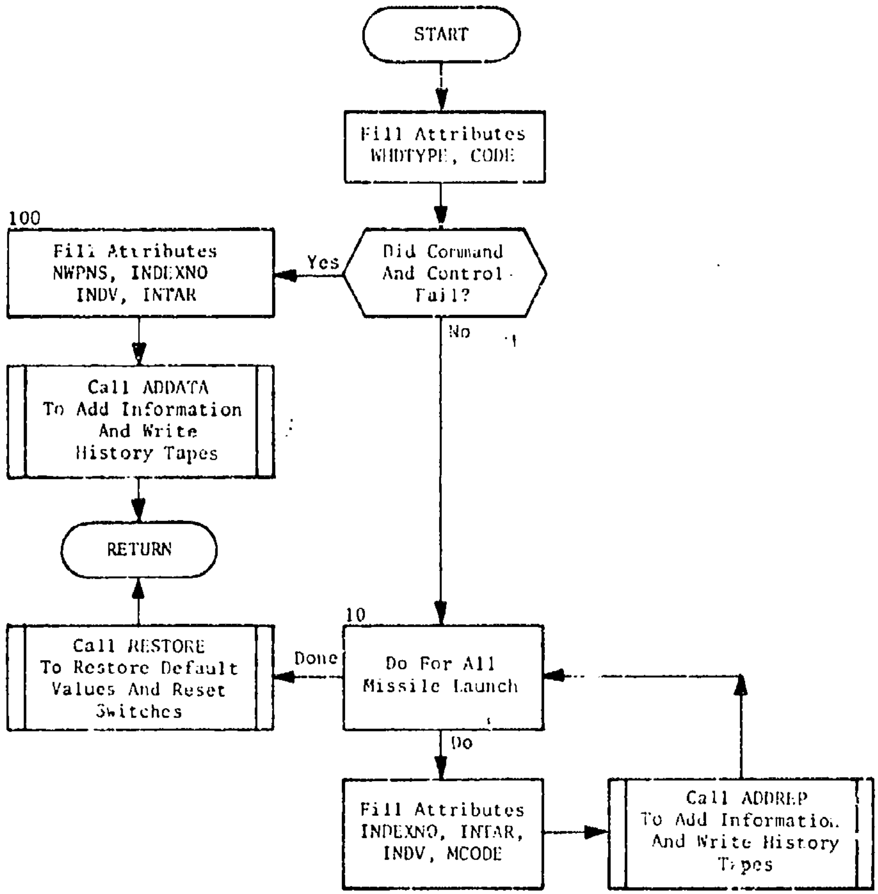


Fig. 75. Subroutine MLAUNX

SUBROUTINE MOREDATA

PURPOSE: To retrieve information not available to the Simulator from the indexed data base (INDEXDB).

ENTRY POINTS: MOREDATA

FORMAL PARAMETERS: None

COMMON BLOCKS: ASM, BCDKLS, BOMBER, BYTYPE, EDITAPE, EDITERM, ITP, KEYS, 1, 2, 3, MAXES, MISSILE, MYIDENT, PROCESS, TAPES, TWORD, WARHEAD

SUBROUTINES CALLED: ABORT, INITEDIT, INPITEM, IPUT, KEYMAKE, NEWUNIT, NEXTITEM, SETREAD, SKIPFILE, TERMTAPE

CALLED BY: READSUM

Method

MOREDATA calls function KEYMAKE to set up keys for packing latitude and longitude, and the indices to arrays storing CNTRYOWN, CNTRYLOC, FUNCTION, and IREG. Subroutine SKIPFILE is called to position INDEXDB to the file containing the breakpoint tables. These are buffered in, and INDEXDB is rewound. The directory is then read from INDEXDB and copied to HISTDATA and HISTAGZ. The switch NOUT is set so that nothing further will be copied to them during the processing of INDEXDB and so that they will not be terminated at the conclusion of the reading of this tape.

As the reading of INDEXDB is done, several arrays are filled: the class tables relating the class name to the value of ICLASS, and the attributes YIELD, FFRAC, and PDUD for items in class WARHEAD, indexed by WHDTYPE. For ASMs, bombers, and missile sites for which ISITE = 1, the CEP is placed in arrays CEPA, CEPB, and CEPM (statements 11, 21, 31), respectively. Items in CEPA are indexed by ASMTYPE; items in CEPB and CEPM are indexed by JTYPE.

For all items which have an INDEXNO, latitude and longitude are scaled by 100000., converted to integers and packed into one word through calls on IPUT (statement 6040). This word is checked to see whether it already appears in KLATPK. If it does not, it is added to the array (statement 6001). In either case, the index of its position in KLATPK is packed into NDXCTY, indexed as are all the following by INDEXNO. The same method is used to store the value of CNTRYOWN and CNTRYLOC in KNTRY, FUNCTION in KFCN, IREG in KREG, and the associated indices in NDXCTY. NDESIG and TASK are packed into one word and written with INDEXNO onto a temporary storage tape.

When the last item on INDEXDB has been processed, the output assignments are restored and control is returned to READSUM.

Subroutine MOREDATA is illustrated in figure 76.

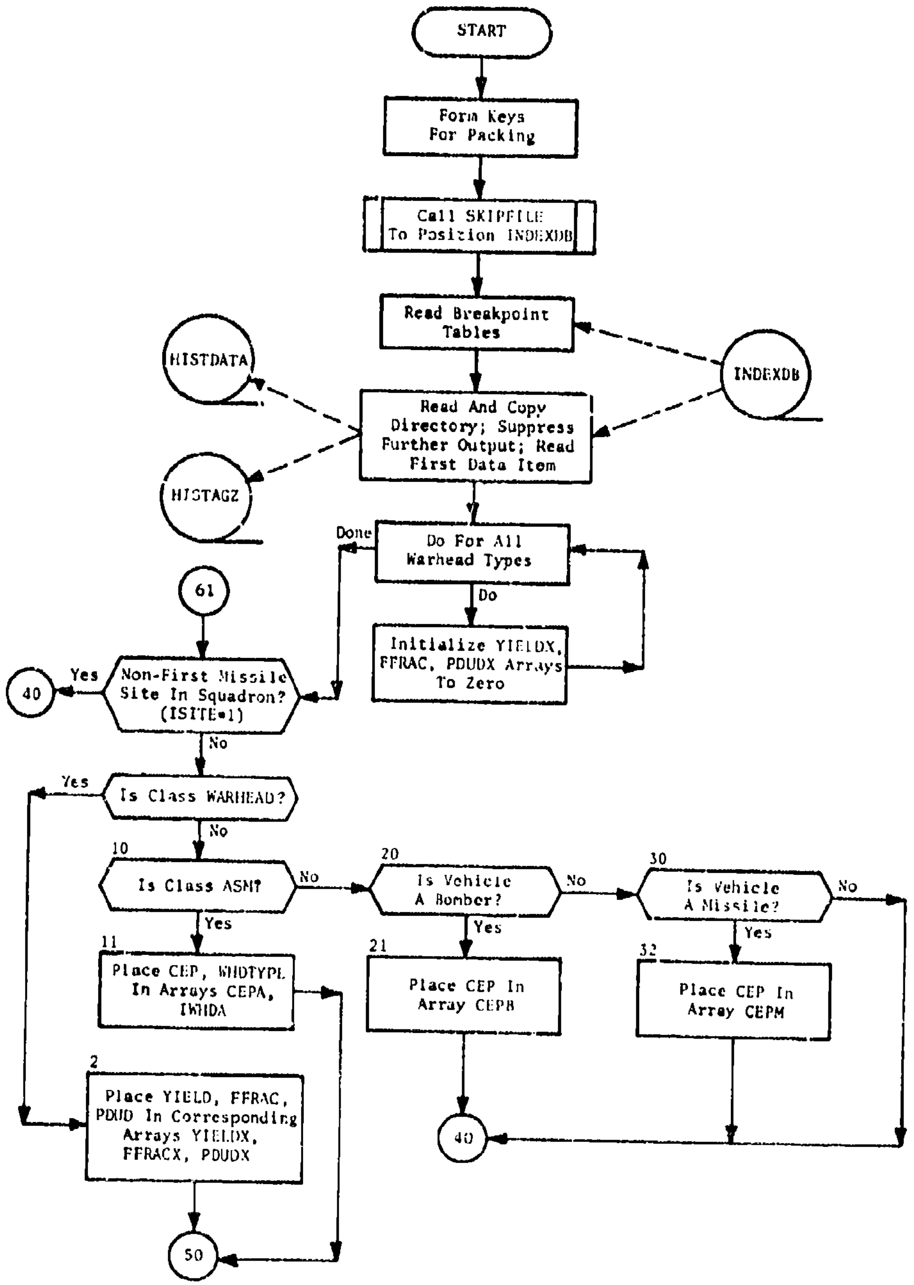


Fig. 76. Subroutine MOREDATA
(Sheet 1 of 2)

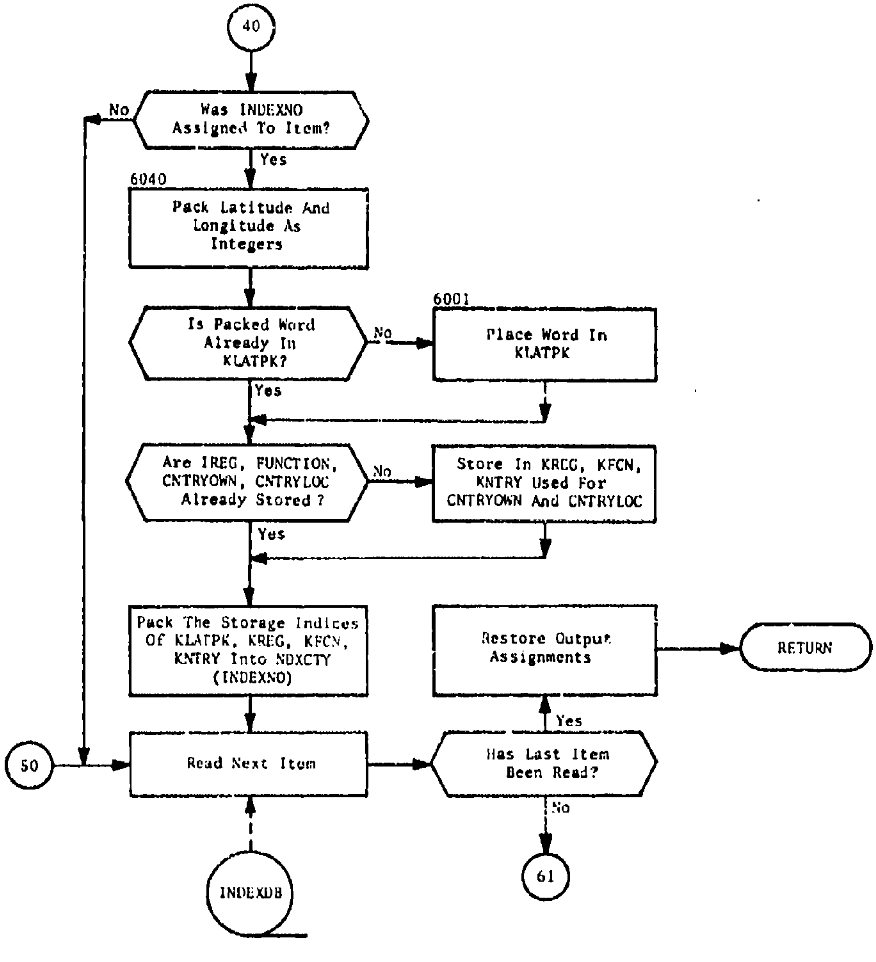


Fig. 76. (cont.)
(Sheet 2 of 2)

SUBROUTINE READATA

PURPOSE:	To read input data cards containing the first four sets of user-input parameters.
ENTRY POINTS:	READATA
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CMCD, DOAGZ, KNTWRT, STIME, WPSYS
SUBROUTINES CALLED:	None
CALLED BY:	READSUM

Method

Subroutine READATA (figure 77) reads four sets of user-input parameters which establish: the game start time; the command/function codes; the weapon system codes; and the parameters which control the writing and printing of the output tapes for REST and SIDAC. The data card formats associated with these parameters are described in chapter 5 of User's Manual, Volume II (see Program READSUM, User-Input Parameters).

The first data card read, five integer fields of 10 columns, contains the game start time: day, month, year, hour, and minutes, in that order.

The command/function codes used in production of the strike tapes AGZBLUSW, AGZBLURO, AGZREDSW, and AGZREDRO are read from the next two data cards. Each card column corresponds to a type delivery vehicle (type refers to the type number as shown in the breakpoint tables output by program INDEXER). This allows the command/function code, an integer value 1 through 9, to be input for a maximum of 160 types.

Next, four data cards are read which contain the weapon system codes to be associated with each command/function code when preparing the strike tapes for REST (AGZREDRO and AGZBLURO). The first two cards contain up to nine weapon system codes (three alphameric characters) for side Blue; the last two contain the codes; e.g., LRA for long-range aviation, for side Red.

The next card read has two fields: the first contains the parameter AGZTAPES if the output tapes for REST and SIDAC are to be written;

the second has the word PRINT if the strike tapes are to be printed. A blank field signifies that the option is not requested. A message is printed reflecting the request.

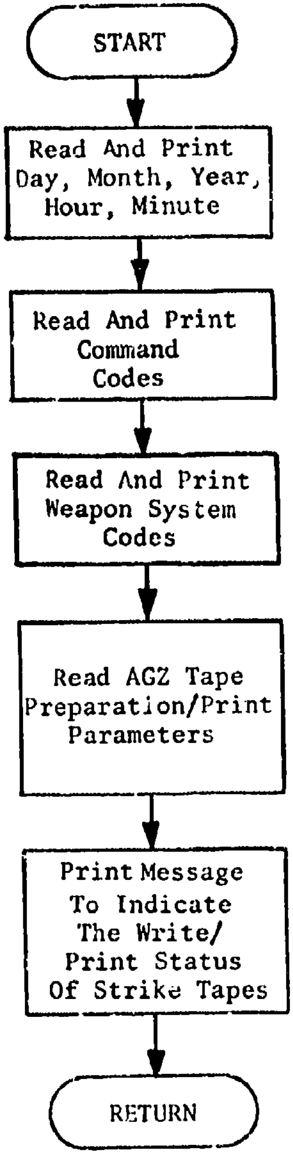


Fig. 77. Subroutine READATA

SUBROUTINE READOUT

PURPOSE:	To transfer information for an event from the HISTAPE to the VALUE array.
ENTRY POINTS:	READOUT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BYTYPE, EDITAPE, EDITERM, HISTOUT, ITP, KNTWRT, PROCESS, ST, TAPES, TWORD
SUBROUTINES CALLED:	ABORT, BDAMX, BOMBANDT, CHANGE, CLAUNX, MATTRITX, MLAUNX, NEWUNIT, PRITEM, RDARRAY, RDWORD, TERMTAP
CALLED BY:	READSUM

Method

READOUT begins by reading the number of words specified for this event on the HISTAPE (statement 100). Since the array (HISTOUT) into which these data will be read can accommodate only 200 items, the number of words for each event is checked. If more than 200 words are specified, information for the last event processed is printed, the first 200 words for this event are read into HISTOUT and printed, and the run is aborted. Otherwise, the information is read into HISTOUT for transfer to the VALUE array. Attributes which are specified for all events, including SIDE, TIME, EVENT, IREG, IALERT, TYPE, CLASS, ICLASS, and ITYPE, immediately are placed in the appropriate attribute in the VALUE array. Then, depending on the type of event being processed (see table 25), one of the following subroutines is called to transfer information which pertains to the kind of event:

MLAUNX	- Missile Launch Event	- Event 1
BOMBANDT	- Bomber and Tanker Events	- Events 2, 4, 5, 7, 8, 11-17, and 19-21
CLAUNX	- Complete Launch Event	- Event 3
MATTRITX	- Missile Attrition Events	- Events 9, 18
BDAMX	- Burst/Damage Event	- Event 10

After control is returned from one of the above subroutines, the reading of the HISTAPE continues. When the last item has been processed, control is returned to READSUM.

Subroutine READOUT is illustrated in figure 78.

Table 25. Event Numbers

EVENT	DESCRIPTION
1	Missile launch
2	Bomber or tanker launch
3	Missile complete launch
4	Bomber refuel
5	Enter zone
6	Zone status
7	Area attrition
8	Local attrition
9	Terminal ballistic missile defense
10	Burst damage
11	Tanker enter refuel area
12	Tanker leave refuel area
13	Bomber or tanker abort
14	ASM launch
15	Decoy launch
16	Recovery
17	Change altitude
18	Area ballistic missile defense
19	Check after recovery
20	Determine time of naval attrition
21	Naval attrition

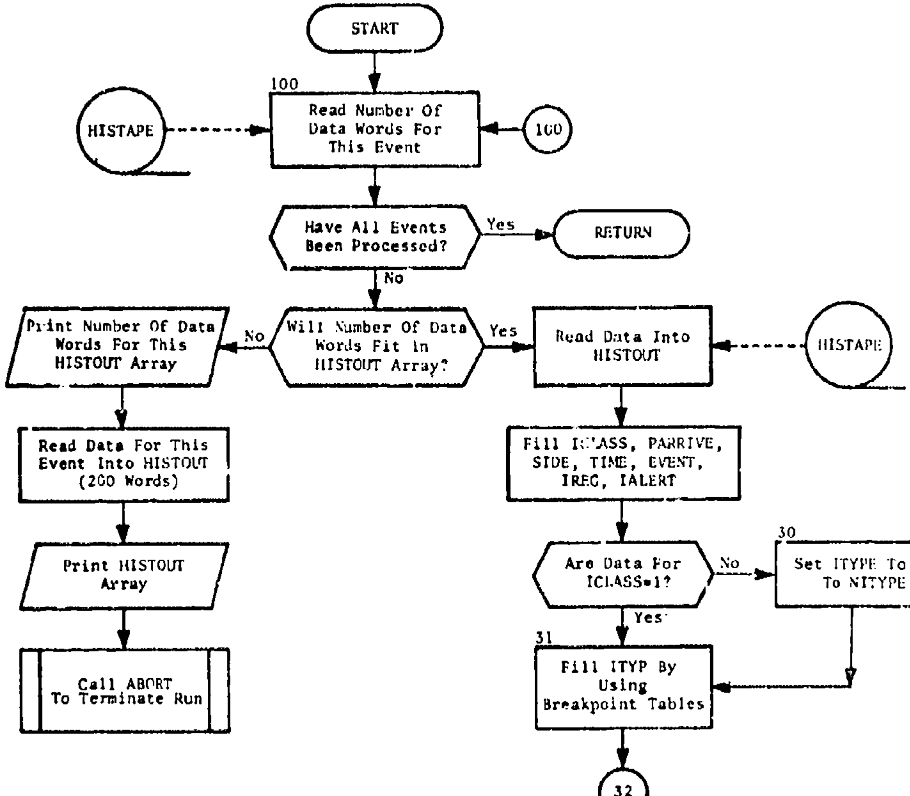


Fig. 78. Subroutine READOUT
(Sheet 1 of 2)

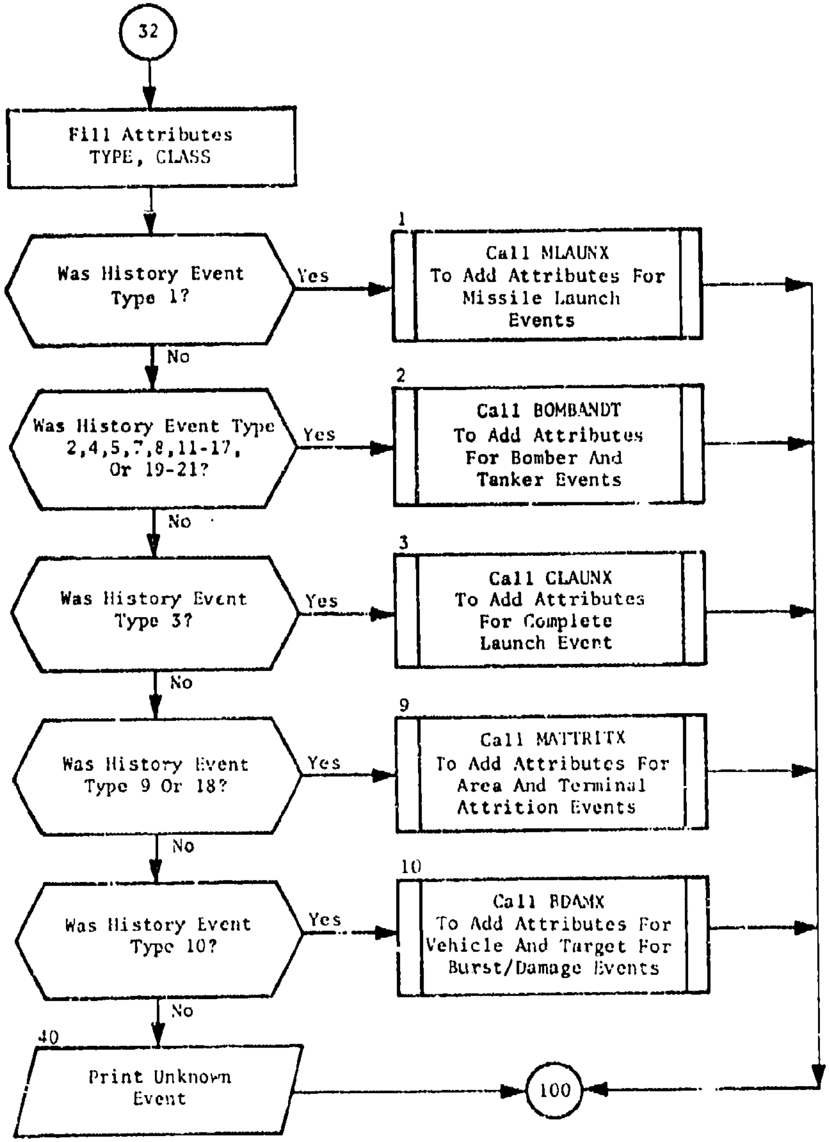


Fig. 78. (cont.)
(Sheet 2 of 2)

SUBROUTINE SPCTBL

PURPOSE:	To process each item which has been written on HISTDATA and update the cumulative AGZ and megaton tables when appropriate.
ENTRY POINTS:	SPCTBL
FORMAL PARAMETERS:	None
COMMON BLOCKS:	EDITAPE, EDITERM, IREGS, MAXES, PROCESS, SPECIAL
SUBROUTINES CALLED:	ABORT
CALLED BY:	GENTBLES

Method

On the first call, the time intervals which become the columns of the tables, and the region names and country group names which become the row labels, are read (statement 5000). Thereafter, this section of the subroutine is bypassed.

If the item being processed is not a prime target (PRIMETAR = 1), control is returned to GENTBLES. Otherwise, for the current side, region, country group, and time, the number of AGZs is incremented and the delivered megatonnage is accumulated in the appropriate entries in the standard summary tables (statement 430).

Subroutine SPCTBL is illustrated in figure 79.

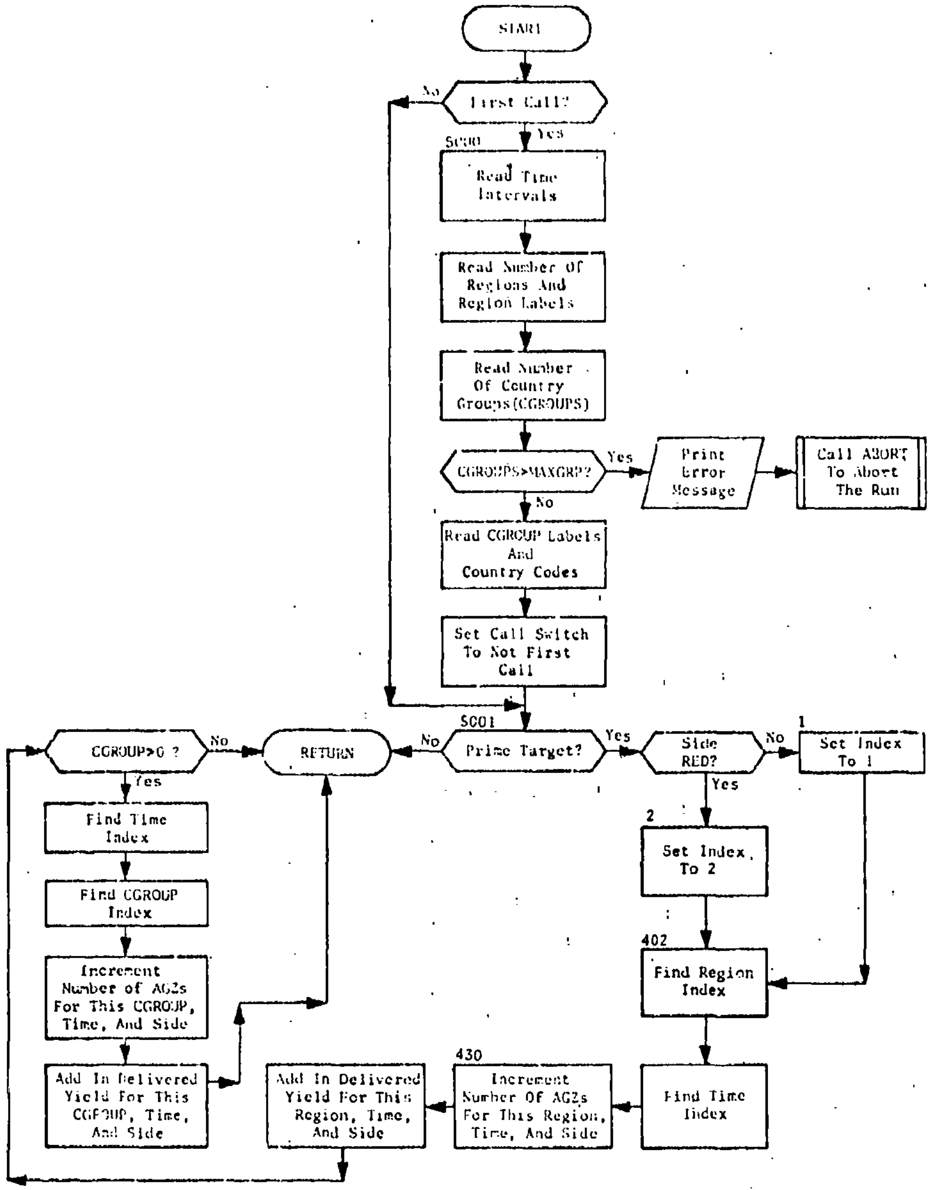


Fig. 79. Subroutine SPCTBL

SUBROUTINE SPCTBLPT

<u>PURPOSE:</u> To print the AGZ and megatonnage standard summaries accumulated by SPCTBL.

<u>ENTRY POINTS:</u> SPCTBLPT

<u>FORMAL PARAMETERS:</u> None

<u>COMMON BLOCKS:</u> SPECIAL

<u>SUBROUTINES CALLED:</u> None

<u>CALLED BY:</u> READSUM

<u>Method</u>

This subroutine prints the tables of data accumulated by subrouti. SPCTBL. These tables by RED and BLUE are the cumulative number of AGZs for various regions, country groups, and time intervals as well as the yield delivered for various regions, country groups, and time intervals.

Subroutine SPCTBLPT is illustrated in figure 80.

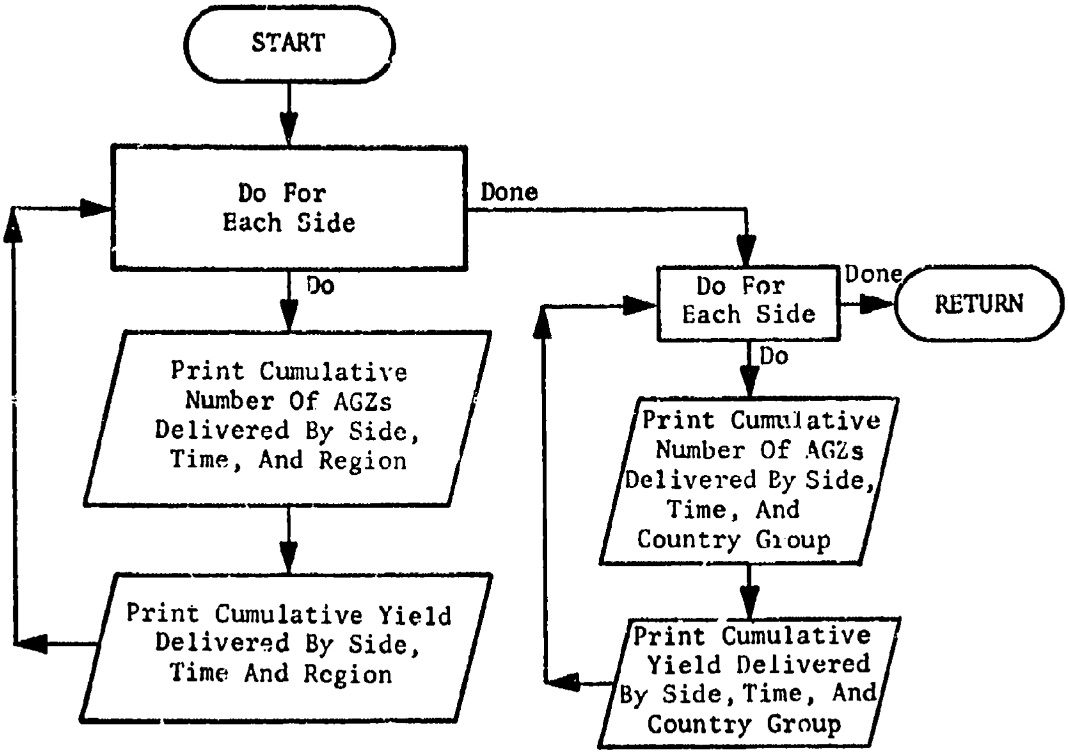


Fig. 80. Subroutine SPCTBLPT

SUBROUTINE WRAGZ

PURPOSE:	To convert data for Burst/Damage events to the form which is required for input to the damage assessment models REST and SIDAC, and to write the associated output tapes.
ENTRY POINTS:	WRAGZ, WRAGZF
FORMAL PARAMETERS:	None
COMMON BLOCKS:	1, 2, 3, ASM, BOMBER, CMCD, DOAGZ, EDITAPE, EDITERM, ITP, KEYS, MAXES, MISSILE, PROCESS, STIME, TAPES, WARHEAD, WPSYS
SUBROUTINES CALLED:	IGET, INZERO
CALLED BY:	BDAMX, READSUM

Method

WRAGZ first examines the user-input parameter to determine if strike tape production has been requested. If it has not, control returns to the calling program, (statement 401). WRAGZ converts the actual ground zero offset distances (AGZ, AGY) from 50ths of miles to degrees of latitude. In order to accomplish this, it is necessary to retrieve the prime target latitude (XLAT) and longitude (XLONG) which were packed by subroutine MOREDATA. After AGZ and AGY have been changed to degrees, they are added to XLAT and XLONG, respectively, to find the actual point on the earth at which the burst occurred (statements 10-38). The appropriate directional labels (north or south, east or west) are assigned, and the coordinates are separated into degrees, minutes, and seconds. The burst also is labeled as being either an air (A) (statement 40) or a ground (G) burst (statement 41).

The time which has elapsed between this Burst/Damage event and the beginning of the game now is computed by adding the current time (TIME) to the game starting time and converting the result to minute, hour, day, month, and year (statements 350 to 199). This information is packed into two words (IDATIME) (statement 222) and processed by function INZERO to replace all blanks with zeros. Similarly, the latitude and longitude data (degrees, minutes, seconds, and direction) and the warhead type index are packed into the words LATX, LONGX, and IWHD, respectively, and are processed by INZERO. The resulting information, together with the strike designation (IFSS), target index (INTAR), and height of burst (A or G) are written onto either AGZBLUS or AGZREDS in card image format (see tables 18 and 19). Control then is returned to BDAMX.

Entry WRAGZF is called once from READSUM to finish writing the strike tapes if this option has been requested. First, DESIG and TASK for each INDEXNO are read from a temporary storage file written by MOREDATA. Then, AGZBLUS and AGZREDS are read, and the strike card images augmented by DESIG and TASK are written to AGZBLUSW and AGZREDSW, respectively. AGZBLUSW and AGZREDSW are then read, the weapon code is added, and AGZBLURO and AGZREDRO are written. The card images are printed if this option has been requested.

Subroutine WRAGZ is illustrated in figure 81.

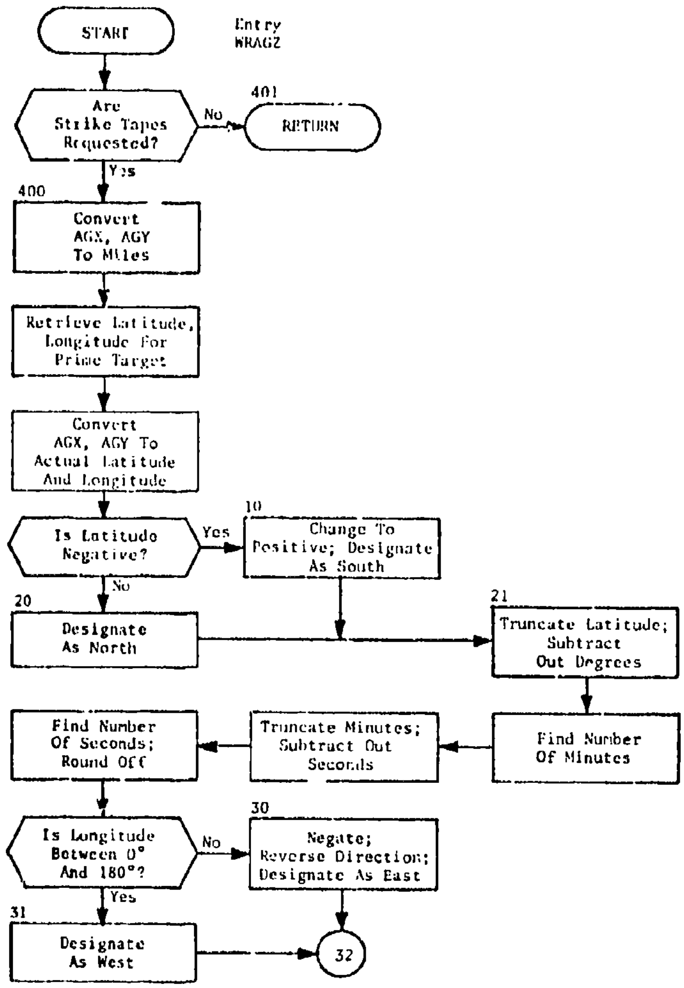


Fig. 81. Subroutine WRAGZ
(Sheet 1 of 4)

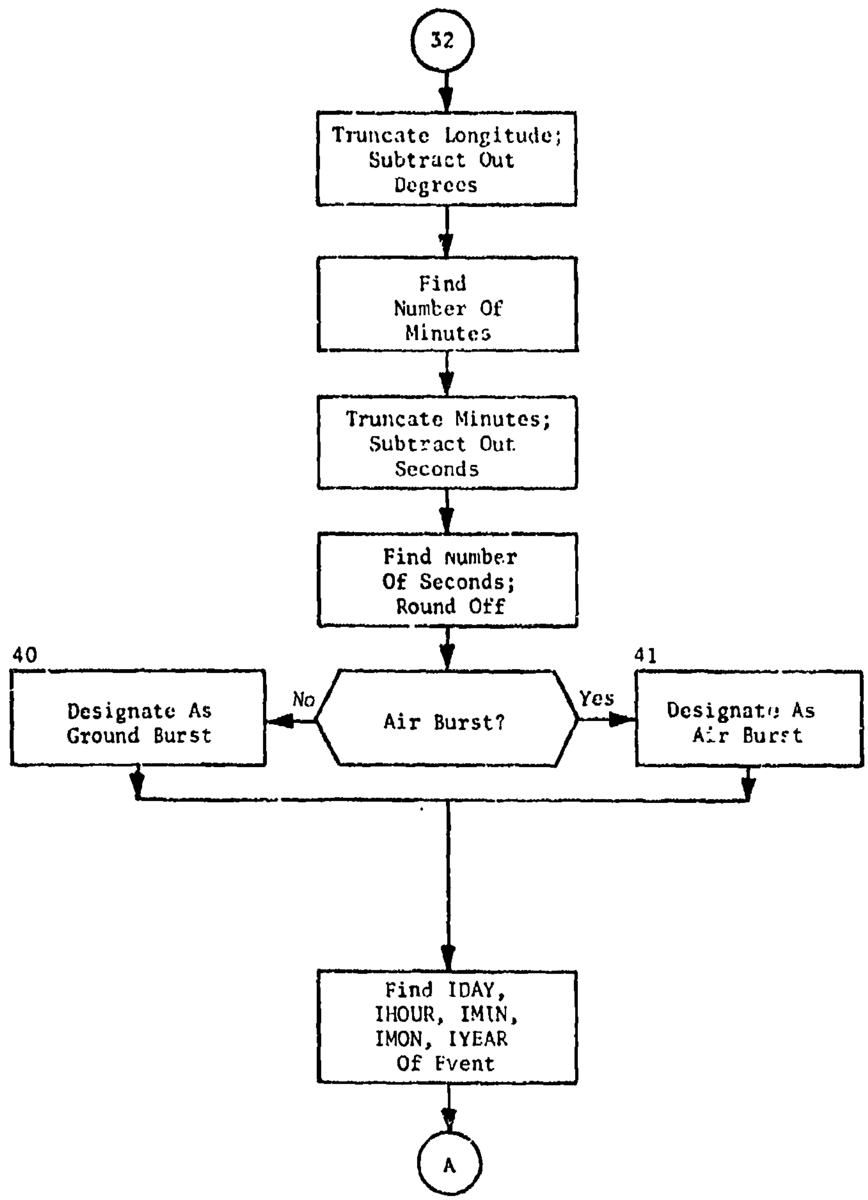


Fig. 81.- (cont.)
(Sheet 2 of 4)

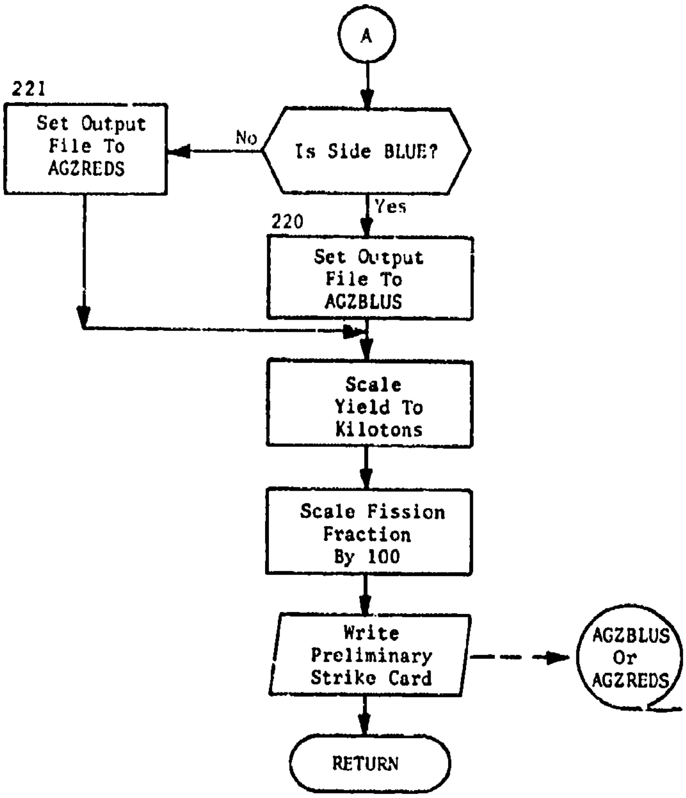


Fig. 81. (cont.)
(Sheet 3 of 4)

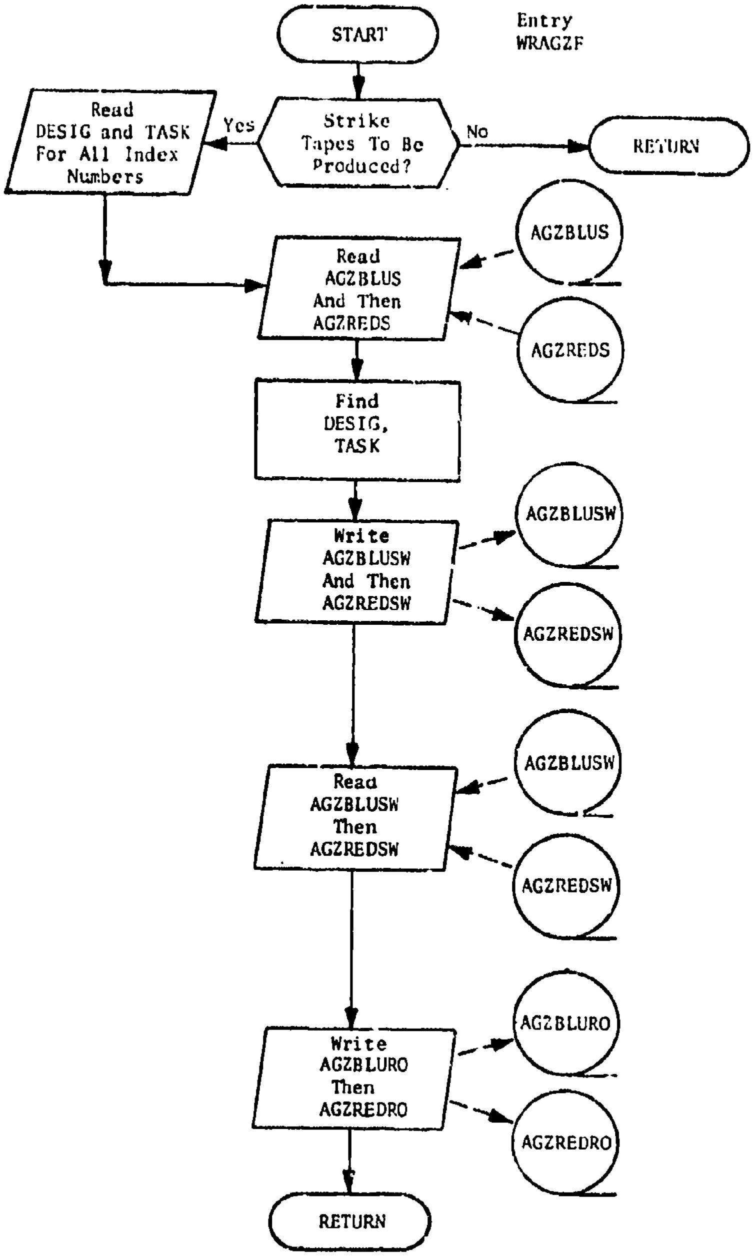


Fig. 81. (cont.)
(Sheet 4 of 4)

CHAPTER 4
PROGRAM TABGEN

PURPOSE

Program TABGEN is a general-purpose report generator which summarizes the HISTDATA or the HISTAGZ tape on the basis of user-input parameters. In contrast with READSUM, the output of the program is not fixed but is as variable as the contents of HISTDATA and the summaries requested by the user can make it. An understanding of the HISTDATA tape and a knowledge of which summaries are meaningful is essential.

It is intended that the user be able to examine in detail only specified portions of the output data base or to generate overall summaries by SIDE, TYPE, or CLASS or any other defined attribute in the data base.

INPUT FILES

TABGEN uses as input either the HISTDATA or the HISTAGZ tape. These tapes are output from program READSUM. HISTDATA is the full output of every simulated event from program SIMULATE; HISTAGZ is a subset of this tape containing only Burst/Damage events.

These tapes have attribute-value pairs filled according to the type of event. These attributes can be counted, added, or used as row or column labels.

The user-input parameters specify the page, row, and columns which are to be summarized as well as other criteria to accept or reject an item.

OUTPUT FILES

There are no ouput files from program TABGEN. The output consists only of the user-specified summary tables.

CONCEPT OF OPERATION

TABGEN allows the user to specify tables to be printed rather than to have standard summaries. However, this choice is at the expense of multipasses through HISTDATA or HISTAGZ. It is also necessary to have a thorough knowledge of the contents of these tapes.

Tables 26 through 31 list the attributes which are defined for each type of event output by SIMULATE. These attributes have been filled in program READSUM by using the NHISTOUT arrays from tape HISTAPE, the breakpoint tables, weapon and payload tables, and other information such as FUNCTION and CNTRYOWN found on INDEXDB.

The user must choose his output page from the list of available attributes. The columns might be ranges of the value of the attribute TIME. The rows could be values of IREG and the values within the table the accumulation of YIELD. Beyond that, TYPE could be used to produce a separate table for each type, and as the last level SIDE could be specified to produce a set of reports for each value of SIDE.

SUMMARY OF SUBROUTINES PERFORMING MAJOR FUNCTIONS

The calling hierarchy of major subroutines in the program is shown in figure 82.

Subroutine INITIL initializes program variables and clears arrays.

Subroutine SETUP reads and interprets all the user-input parameters and sets all the arrays and variables which will control the accumulation of data and the final printing of the pages.

Function INTREL sets a switch on the basis of the user-supplied parameters so that the proper selection logic will be done on the values of attributes in the data base input item.

Subroutine TRUTH examines each item according to the criteria specified by the user and interpreted by SETUP and determines which, if any, tables are affected. It calls NVALFIND to perform the logical comparisons required.

Subroutine CYCLER controls the paging or cycling. It examines the attribute used for paging to determine if the page is an old one or one newly encountered.

Table 26. Attributes Filled for All Events

ICLASS	PARRIVE	IREG	CLASST
TIME	EVENT	CLASS	FDUD
IALERT	TYPE	FUNCTION	CNTRYLOC
ITYPE	SIDE	TYPET	CNTYLOCT
YIELD	CNTRYOWN	CNTYOWNT	LAT
LONG			

Table 27. Attributes Filled for Missile Launch Events

WHDTYPE	CODE	INDEXNO
INTAR	INDV	MCODE/NWPNS*

*MCODE filled when no command and control failure; NWPNS if command and control failure.

Table 28. Attributes Filled for Bomber and Tanker Events

INDEXNO	INDV	ZONE	NDECOYS
IALT	NCM	DELAY	BCODE
IDUD	ASMTYPE	PLACE	IOTHER
TIMEN	EVENTN	ASMTYPE	PLACEN
WHDTYPEN			

Table 29. Attributes Filled for Complete Launch Events

INDEXNO	INDV	INTAR	WHDTYPE
CODE			

Table 30. Attributes Filled for Ballistic Missile Defense Events

NWPNS	NWHDS	INDEXNO	INDV
INTAR	WHDTYPE	NAL	TAIM
CODE	PLACE/NPEN*	NWHDS/NDET	

*PLACE, NWHDS are filled for terminal attrition; NPEN, NDET are filled for area attrition.

Table 31. Attributes Filled for Burst/Damage Events

PRIMETAR	INDEXNO	INDV	WHDTYPE
INTAR	CODE	DGX	DGY
DHOB	AGX	AGY	AHOB

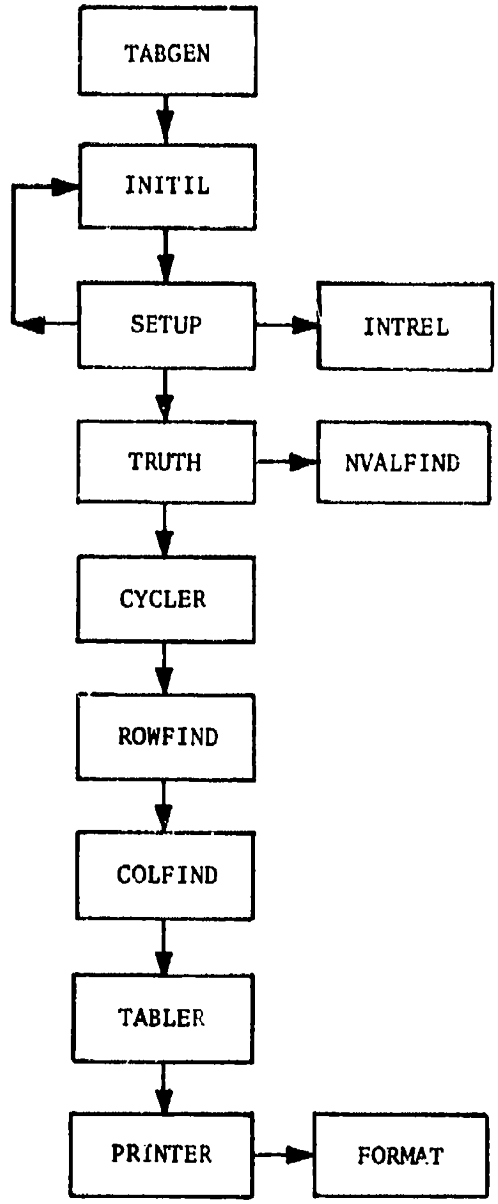


Fig. 82. **Calling Hierarchy of Program TABGEN**

Subroutine ROWFIND allows either a prespecified list of rows or a dynamically generated one. In the first, an item is placed in the proper row if there is a corresponding value of the row attribute in the table. In the second, a new row is generated whenever a new value of the row attribute is encountered.

Subroutine COLFIND determines whether an item belongs in a particular column according to one of three user-specified criteria: the value of the column attribute must be a member of a list; the value of the attribute may be placed in the list if not already present; or the value is compared with lists of ranges of values to find the proper column.

If an item has been found to belong to a page, row, and column, that particular entry is incremented either by one or by the actual value of the attribute according to the user-specified parameters.

Subroutine PRINTER prints all of the summary tables.

COMMON BLOCK DEFINITIONS

External Common Blocks

External common blocks are described in the Programming Specifications Manual, Volume I, Data Input Subsystem, Part A. They are /EDITAPE/, /EDITERM/, /PROCESS/, /ITP/, and /DIRECTRY/.

Internal Common Blocks

The internal common blocks used by program TABGEN are shown in table 32.

Table 32. Program TABGEN Internal Common Blocks (Sheet 1 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
001	TABLE(NTABLE)	Storage of all table entries
	XVAL(MVAL)	Cell to allow fixed or floating addition into table
ATTIND	ICYCLE	Index of paging attribute in directory
	IROW	Index of row attribute
	IADD	Index of tabled attribute
	ICOL	Index of column attribute
	JSIDE	Index of attribute SIDE
COLS	NCOL	Number of column identifiers, if specified (attribute values)
	RANGES	List of allowed attribute values or range of values
COLVAR	MCOL	Number of columns on each logical page
	NAMECOL	Name of each column for each logical page
COMMAND	NCOMM	Number of commands recognized
	ICOMMAND	List of commands
CYCS	NAMECYC	Value of page attribute for each cycle
	NUMCYC	Number of cycles
	CYCSIDE	Value of SIDE for each cycle
FORMATS	INWORD	INPUT number
	NFORMT	Eight-column format
	INABS	Absolute value of INWORD
	IF2X	Set to 1 to augment NFORMT with 2X

Table 32. (cont.)
(Sheet 2 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
HEAD	NHEADQ	Number of page titles
	HEAD	Page titles
INITROW	NUMROWS	Number of allowed row values if specified
	INITROW	List of allowed row values
	MXROW	Maximum number of rows
ITIME	ITIME	Index of attribute TIME
	NTEMP	Temporary storage location
JXCODE	TABCODE	Indicates fixed or floating values in table
	IFWHAT	Indicates whether or not user has specified format of table values
JXFORM	ROWFORM	Format of row attribute
	TABFORM	Format of table attribute
	COLFORM	Format of column attribute
	CYCFORM	Format of paging attribute
PASS	NSIDENOW	Current side if two passes required
	NPASS	Number of current pass
ROWS	NAMEROW	Attribute values for all rows for all cycles
	NUMROW	Number of rows for each cycle
SETS	SETNAME	Set identifier

Table 32. (cont.)
(Sheet 3 of 4)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
SETS (cont.)	SETREL	Logical operator for each set
	SETLEFT	Left-hand side of logical comparison, attribute value, or SETNAME
	VALORSET	Switch to indicate attribute-value comparison or previously calculated set comparison
	SETVAL	Truth value of each set
	SETTRUE	Indicates whether or not set must have value true
	SETDEF	Indicates which sets apply to current table
	NSETS	Number of sets
SIZES	MAXCOL	Maximum number of columns
	MAXROW	Maximum number of rows
	MAXCYC	Maximum number of cycles
	MAXSET	Maximum number of sets
SWITCH	COLINSW	True if item fits a column
	ROWINSW	True if item fits a row
	CYCINSW	True if item fits a page
	TRUESW	True if item meets set definitions
	MAXROWSW	True if row size limit exceeded
	MAXCYCSW	True if cycle size limit exceeded
	MAXCOLSW	True if column size limit exceeded
	ADDSW	1 if values accumulated in table; 2 if incremented by one
	ROWSW	1 if list of allowed row values; 2 if not
	NOSIDESW	1 if sides separated; 2 if not

Table 32. (cont.)
(Sheet 4 of 4

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
SWITCH (cont.)	VALCOLSW	1 if allowed list of values; 2 if range of values given; 3 if all values accepted
	NSEPSW	1 if one pass required; 2 if two
TABINDEX	COLIND	Column in which item belongs
	CYCIND	Cycle in which item belongs
	ROWIND	Row in which item belongs

PROGRAM TABGEN

PURPOSE:	To call all subroutines required to produce and print summary tables on the basis of user-specified input, and to print any error messages.
ENTRY POINTS:	TABGEN
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ATTIND, EDITAPE, EDITERM, ITIME, ITP, PASS, PROCESS, SIZES, SWITCH
SUBROUTINES CALLED:	COLFIND, CYCLER, INITEDIT, INITIL, INITAPE, INPITEM, NEXTITEM, PAGESKP, PRINTER, ROWFIND, SETREAD, SETUP, TABLER, TERMTAPE, TRUTH
CALLED BY:	None

Method

Program TABGEN, on the first pass, calls subroutine SETUP (statement 6) to read and interpret all user-request cards. If a second pass is required to print the RED side, this call is bypassed and subroutine INITIL (statement 7) is called to do the required initialization.

As each item is read from HISTDATA or HISTAGZ, subroutine TRUTH (statement 505) is called to see whether the item fits the most restrictive criteria (such as belonging to the current side or being the requested type of event or vehicle). If the item is not rejected (TRUESW=2), subroutines CYCLER, ROWFIND, and COLFIND (statements 503, 500, 502) are called and the item is rejected if it does not belong on any page, in any row, or in any column. If it is not rejected, subroutine TABLER (statement 504) accumulates the value of the specified attribute in the table, and the next item on the tape is read.

Whenever it is determined that an item does not belong in the table, the program immediately bypasses further examination and reads the next item from tape (statement 501).

When all items have been examined, subroutine PRINTER prints all pages and all tables. A list of error switches are queried which indicate whether or not the user specified more pages, columns, or rows than the program is dimensioned for (statement 600), and appropriate error messages (statements 603, 604, 605) are printed if any table overflow occurred.

If sides had to be done in separate passes through the input tape due to memory limitations, the above process is repeated for the new side (statement 2). Otherwise, the program goes on to the next set of user-specified tables (statement 5) until the signal to terminate is encountered in the call to subroutine SETUP.

Program TABGEN is illustrated in figure 83.

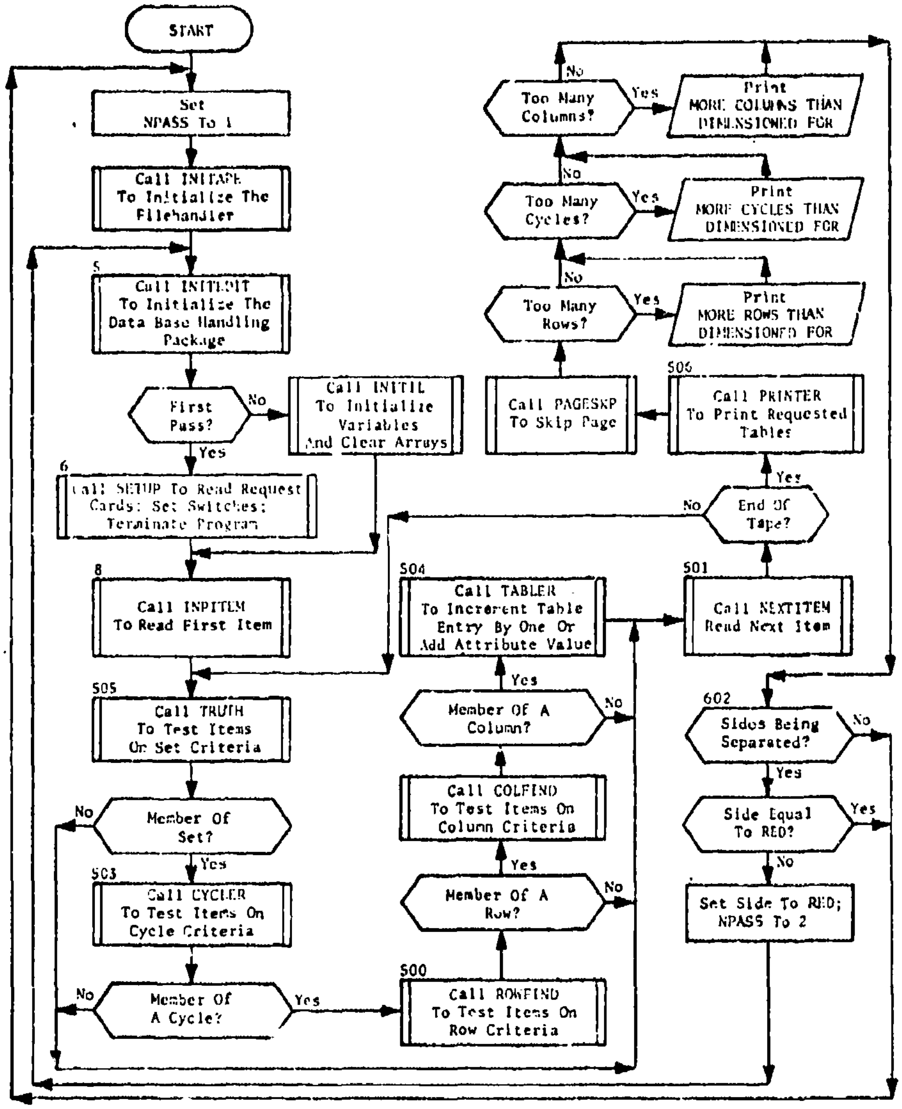


Fig. 83. Program TABGEN

SUBROUTIN COLFIND

PURPOSE: To determine whether the column attribute mat s the column specifications.

ENTRY POINTS: COLFIND

FORMAL PARAMETERS: None

COMMON BLOCKS: ATTIND, COLS, COLVAR, PROCESS, SIZES, SWITCH, TABINDEX

SUBROUTINES CALLED: ITLE

CALLED BY: TABGEN

Method

There are three ways in which the column index may be determined. If a list of values of the column attribute has been input, an exact match must be found (statement 1). If any value of the attribute is acceptable, the index will be set if a match is found among the values already within the list; otherwise, the new value is added to the list and its index used (statements 10 to the RETURN). If a numeric range of values for the columns has been specified, the current attribute value is compared with each limit to determine the correct index (statement 2 and beyond).

Statement 20 is encountered whenever a match is found in the restricted list of values. It determines whether a particular page has already encountered this value. If it has not, the value is added to those for that page.

Subroutine COLFIND is illustrated in figure 84.

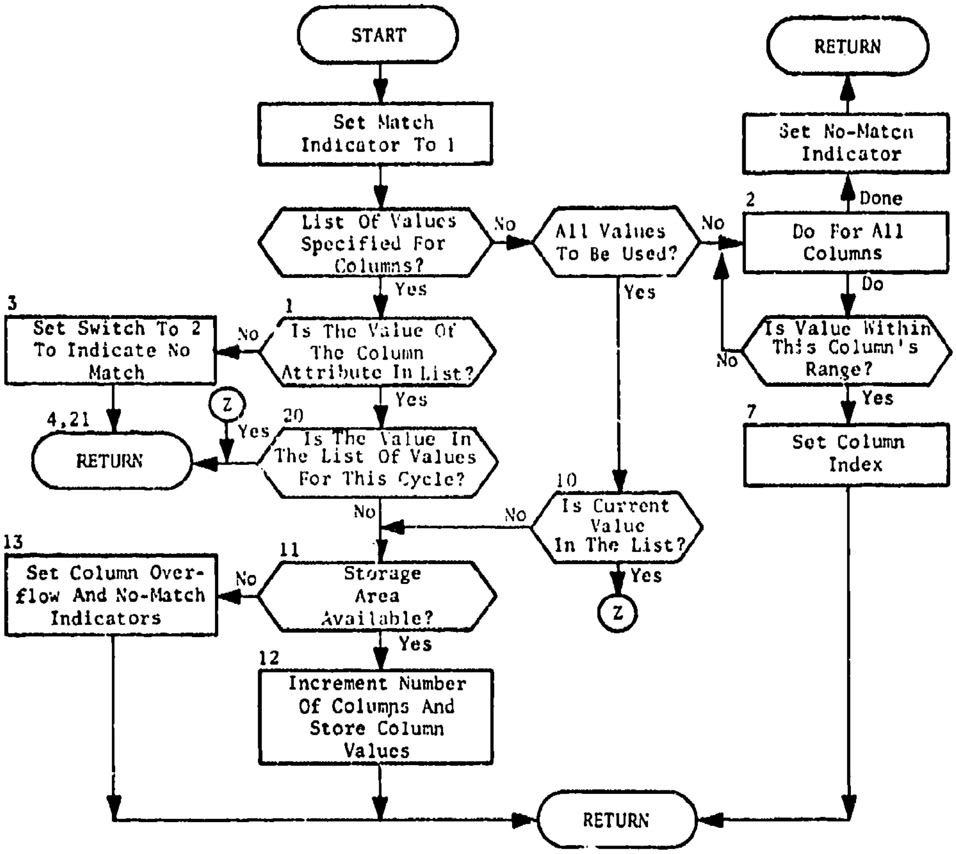


Fig. 84. Subroutine COLFIND

SUBROUTINE CYCLER

PURPOSE: To determine the value of the page index on the basis of the cycling attribute.

ENTRY POINTS: CYCLER

FORMAL PARAMETERS: None

COMMON BLOCKS: ATTIND, CYCS, PASS, PROCESS, SIZES, SWITCH, TABINDEX

SUBROUTINES CALLED: ITLE

CALLED BY: TABGEN

Method

If paging on an attribute is required, this subroutine sets the proper index. If it is not, the index is set to one, or to one or two, if separation of sides has been requested.

The subroutine starts by interrogating ICYCLE to determine if paging is desired. If it is not, CYCIND. the page index, is set to one (statement 8). If separation by side is necessary, the value of SIDE is examined and if it is RED, CYCIND is set to two (statement 12).

When paging on a selected attribute has been requested (statement 9) as well as separation by SIDE, NSEPSW is examined to find out whether both sides can coexist in memory. If not, CYCINSW is set to two if it is not the current side and control returns to TABGEN (statement 21).

When sides are not separated, when both sides can coexist in memory, or it is the current side (statement 17), the current value of the paging attribute is looked up in the list of previously found values. If it is there for the current side, the value of CYCIND is set (statements 15 through 14), and the program returns.

If it is not there (statement 1) and there is not enough room in the list to store the current value, the error switch is set and control returns (statement 6). If there is room, the current value of the attribute and the current value of SIDE are stored, and that value of CYCIND is returned.

Subroutine CYCLER is illustrated in figure 85.

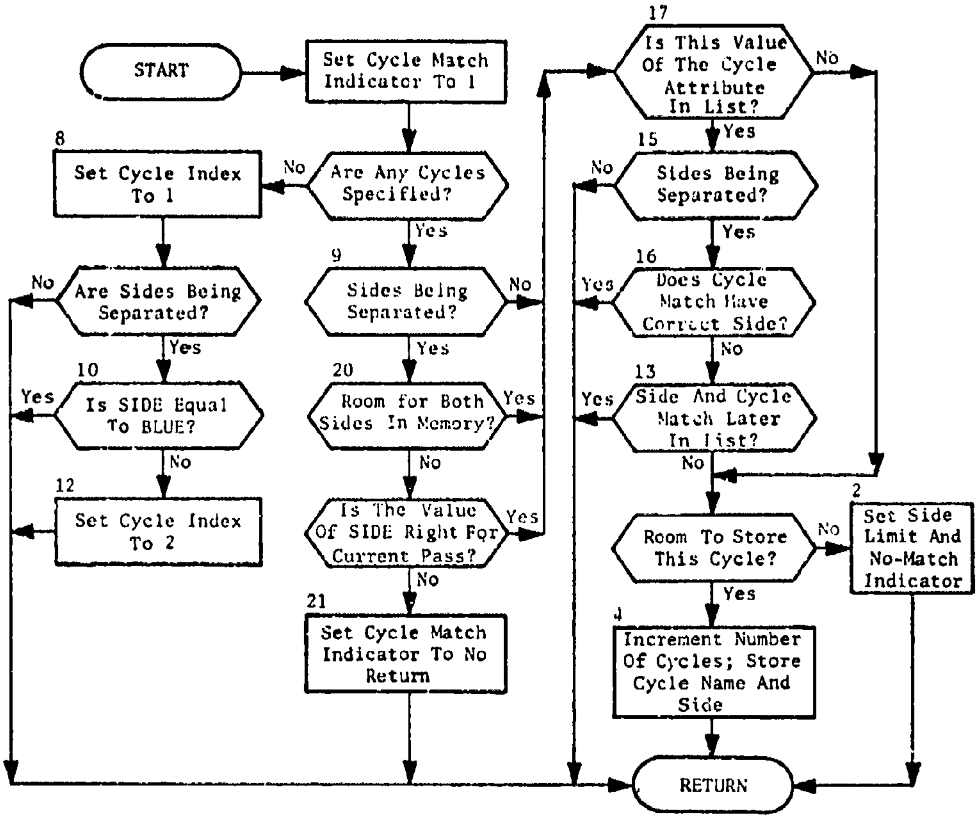


Fig. 85. Subroutine CYCLER

SUBROUTINE FORMAT

PURPOSE:	To determine an appropriate print format for any floating point number.
ENTRY POINTS:	FORMAT
FORMAL PARAMETERS:	None
COMMON BLOCKS:	FORMATS
SUBROUTINES CALLED:	None
CALLED BY:	PRINTER

Method

Since the column width for all tables output from TABGEN has been set to eight, this subroutine determines which print format will allow the number to print.

The absolute value of the number is compared with the largest magnitude which can be printed by several formats (statements 4 through 9). After the appropriate range is found, NFORMT is set to the indicated format (statements 11 through 18).

If the number to be printed is negative, the number of significant digits is reduced by one (statement 20). If the page has room to allow spacing between columns, these spaces are added to the format (statement 21).

Subroutine FORMAT is illustrated in figure 86.

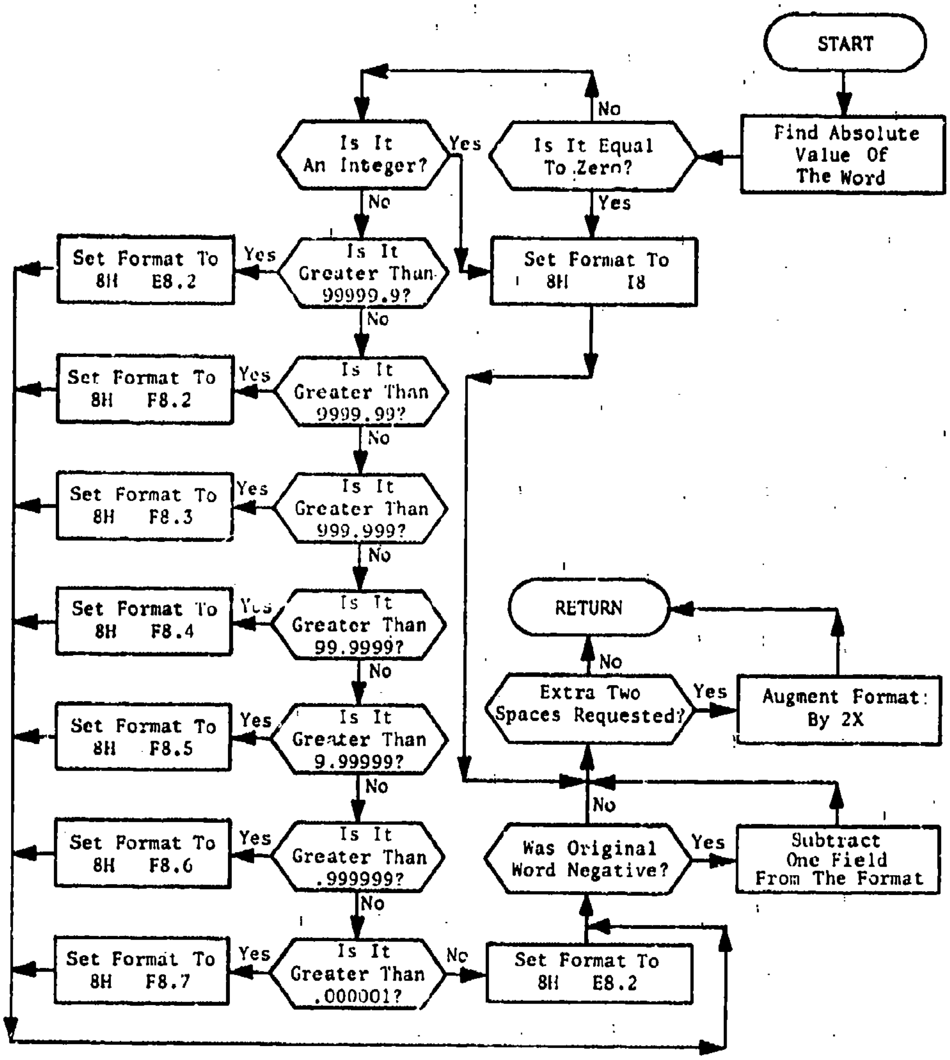


Fig. 86. Subroutine FORMAT

SUBROUTINE INITIL

PURPOSE: To initialize program variables and clear arrays.

ENTRY POINTS: INITIL

FORMAL PARAMETERS: None

COMMON BLOCKS: 001, ATTIND, COLVAR, COLS, CYCS, HEAD, INITROW, PASS, ROWS, SETS, SIZES, SWITCH, TABINDEX

SUBROUTINES CALLED: None

CALLED BY: SETUP, TABGEN

Method

On the first pass through the data, constants and size limits are set and array clearing is performed. On the second pass, only selected arrays and constants are cleared in order to preserve user-input parameters.

Subroutine INITIL is illustrated in figure 87.

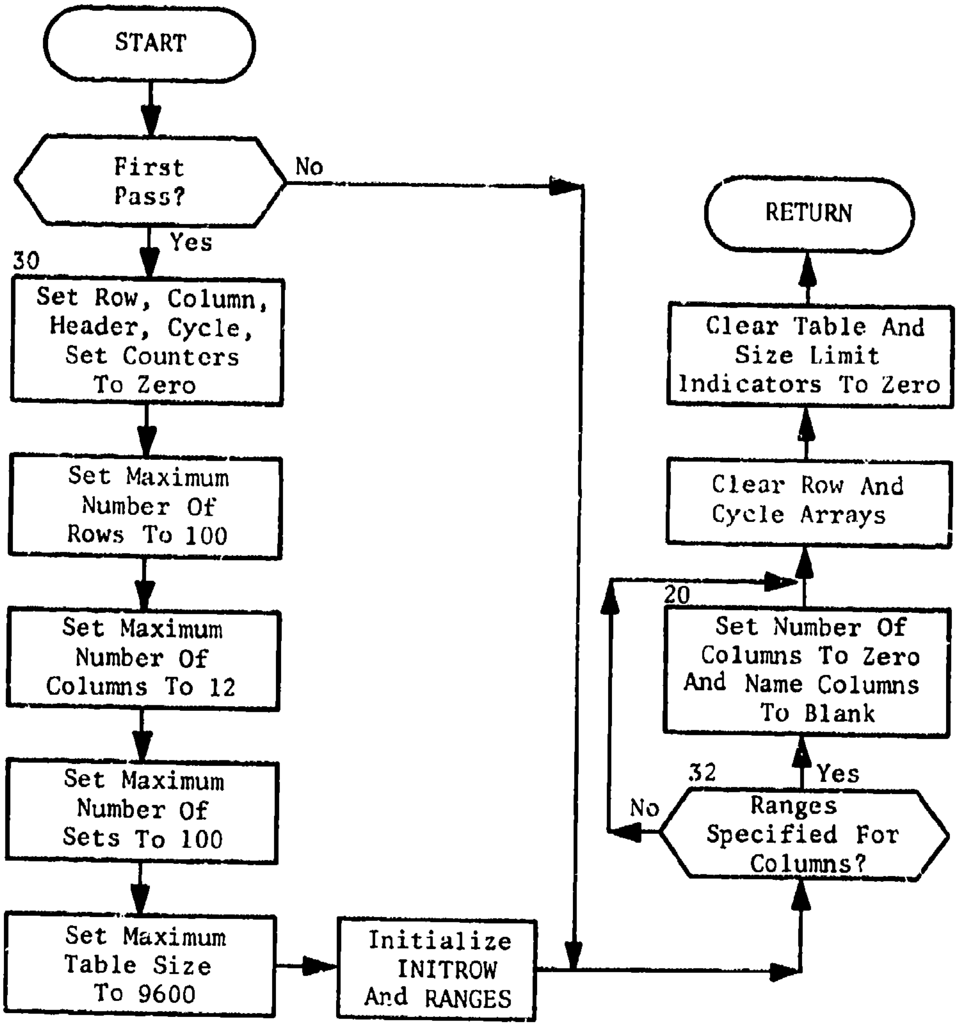


Fig. 87. Subroutine INITIL

FUNCTION INTREL

PURPOSE:	To set a switch to indicate numerically the proper logical comparisons to be made in subroutine TRUTH.
ENTRY POINTS:	INTREL
FORMAL PARAMETERS:	INREL - Alphabetic representation of a logical comparison
COMMON BLOCKS:	None
SUBROUTINES CALLED:	ITLE
CALLED BY:	SETUP

Method

A table of the logical comparisons which may be called for in the user-input parameters is stored by means of a data statement. Each user input is then compared with this list, and the index of the match is returned. A zero is returned if there is no match.

The index is used in function NVALFIND to find the called-for logical comparison.

Function INTREL is illustrated in figure 88.

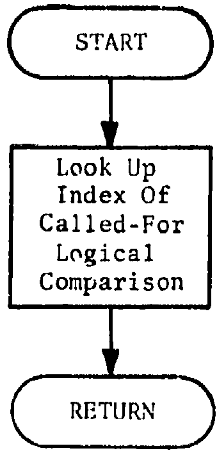


Fig. 88. Function INTREL

FUNCTION NVALFIND

PURPOSE:	To perform logical comparisons and return a true or false value.
ENTRY POINTS:	NVALFIND
FORMAL PARAMETERS:	ISW - A switch to indicate which comparison to perform NP1 - Left side of comparison NP2 - Right side of comparison
COMMON BLOCKS:	None
SUBROUTINES CALLED:	None
CALLED BY:	TRUTH

Method:

ISW acts as a pointer to the appropriate "if" statement (statements 1 through 10). The comparison is made and a one is returned if the statement is true (statement 11) and a zero if it is false (statement 12).

Function NVALFIND is illustrated in figure 89.

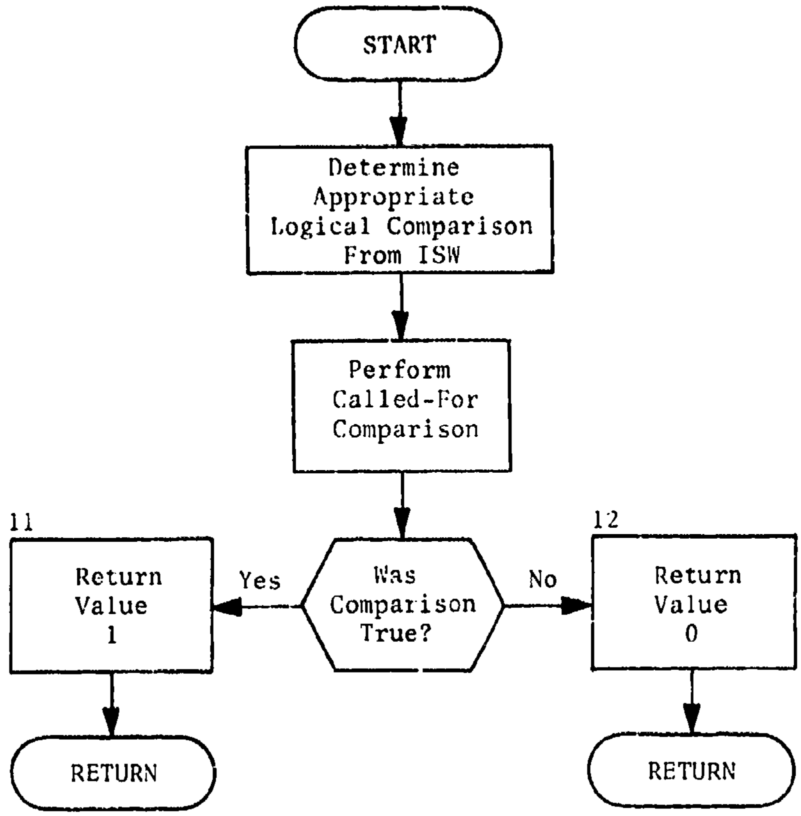


Fig. 89. Function NVALFIND

SUBROUTINE PRINTER

<u>PURPOSE:</u> To print the requested summary tables.

<u>ENTRY POINTS:</u> PRINTER

<u>FORMAL PARAMETERS:</u> None

<u>COMMON BLOCKS:</u> 001, ATTIND, COLS, COLVAR, CYCS, DIRECTRY, FORMATS, HEAD, JXCODE, JXFORM, ROWS, SIZES, SWITCH, TABINDEX

<u>SUBROUTINES CALLED:</u> FORMAT, PAGESKP

<u>CALLED BY:</u> TABGEN

Method

This subroutine prints the user-requested tables. It starts by calculating field widths according to the number of columns in the table and stores format information (statement 5). If there is a paging attribute, it is printed (statement 25). If the printing is to be done by side, the current value of SIDE is printed (statement 39).

The column headings are then printed (statements 35 to 34). The column total array is cleared (statement 62) as well as the row total slot. Each row is then printed in the called-for format with the row label, columns of data, and the row total (statements 10 through 7). After all rows have been printed, the column totals are printed (statements 55 through 54).

After all pages of the report have been printed, column totals over all pages are printed (statements 8 through 100).

Subroutine PRINTER is illustrated in figure 90.

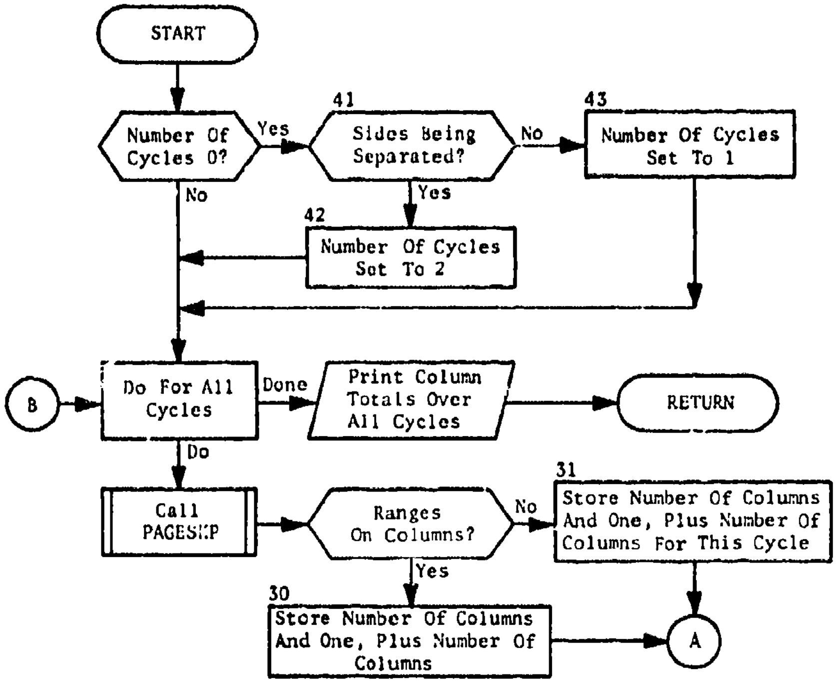


Fig. 90. Subroutine PRINTER
(Sheet 1 of 2)

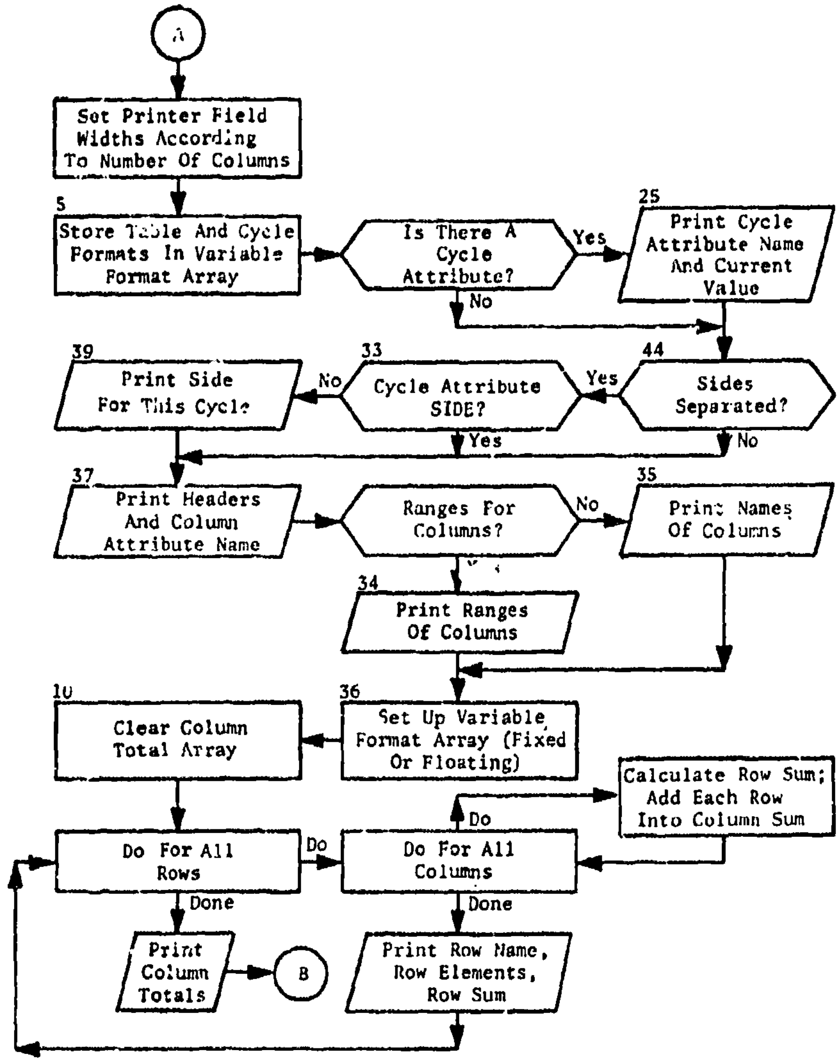


Fig. 90. (cont.)
(Sheet 2 of 2)

SUBROUTINE ROWFIND

PURPOSE:	To find the row matching the current value of the row attribute.
ENTRY POINTS:	ROWFIND
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ATTIND, INITROW, PROCESS, ROWS, SIZES, SWITCH, TABINDEX
SUBROUTINES CALLED:	ITLE
CALLED BY:	TABGEN

Method

The value of the row attribute is examined in one of two ways to determine whether an item fits a row definition. The row attribute is matched against a list of values already appearing on the current page. If there has been a list of preset values and there was no match, the list of preset values is compared to see if there is a match there (statement 12). If not, control returns to TABGEN with a row index (ROWINSW) of two (statement 5). If a match is found, the value is added to the current page, and control returns with ROWINSW set to one (statements 11, 6).

If no preset list has been given, the number-of-rows counter is incremented, the new value is stored (statement 4), and control returns with ROWINSW equal to one.

Subroutine ROWFIND is illustrated in figure 91.

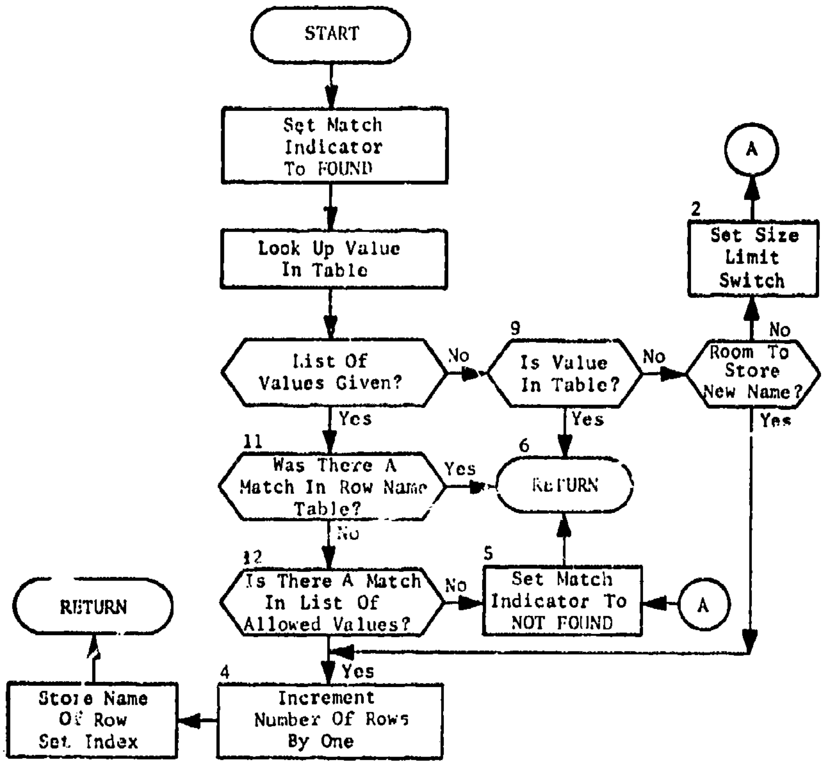


Fig. 91. Subroutine ROWFIND

SUBROUTINE SETUP

PURPOSE:	To read user-input parameters and interpret them; to fill input arrays; and to set internal switches on the basis of these parameters.
ENTRY POINTS:	SETUP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	ATTIND, COLS, COMMAND, DIRECTRY, HEAD, INITROW, ITIME, JXCODE, JXFORM, PASS, ROWS, SETS, SIZES, SWITCH
SUBROUTINES CALLED:	INITIL, INTREL, ITLE, NUMGET, PAGESKP
CALLED BY:	TABGEN

Method

This subroutine reads and interprets user-request cards. There are nine commands recognized. One of them, SIZE, must be the first card, but the rest may appear in any order. If SIZE is not used, 30 columns and 26 rows are the maximum (statements 999 through 303). SIZE is used to vary this limit according to the quality of the data base. If necessary, two passes will be performed, one for side BLUE and one for side RED (statements 304 through 306).

SET is used to extract only certain items from the data base for inclusion in the table. Sets may be of the form SET1 ATTRIBUTE EQ VALUE or SET1 SETA AND SETB where SETA and SETB have been previously defined as in the first form (statements 1 through 16).

HEADER allows input of as many as five cards of labeling to appear at the top of each table (statements 50 through 53).

The CYCLE command may contain an attribute name whose values are to be cycled through and/or the word NOSIDE which indicates that items are to be included regardless of side (statements 60 through 63).

ADD has two options. If the second field is blank, the table entries are incremented by one for each acceptable item. If the second field contains the name of an attribute, the values of that attribute are accumulated in the table. Normally, the format will be taken from the data base directory, but a user-specified one may be placed in the third field of this card (statements 70 through 81).

ROW is followed in the second field by the name of an attribute. If the third field is blank, all values of the attribute will be used; if not, a list of allowed values will be stored and the row names will be restricted to these (statements 90 through 95).

COLUMN has three types of specifications. The first two are comparable to those described above under ROW. The third allows the user to specify a range of values of the attribute for each column (statements 200 through 212).

RUN causes the present set of tables to be generated and printed. New table definitions may follow (statements 230 through 233).

STOP causes the program to terminate (statement 240).

Subroutine SETUP is illustrated in figure 92.

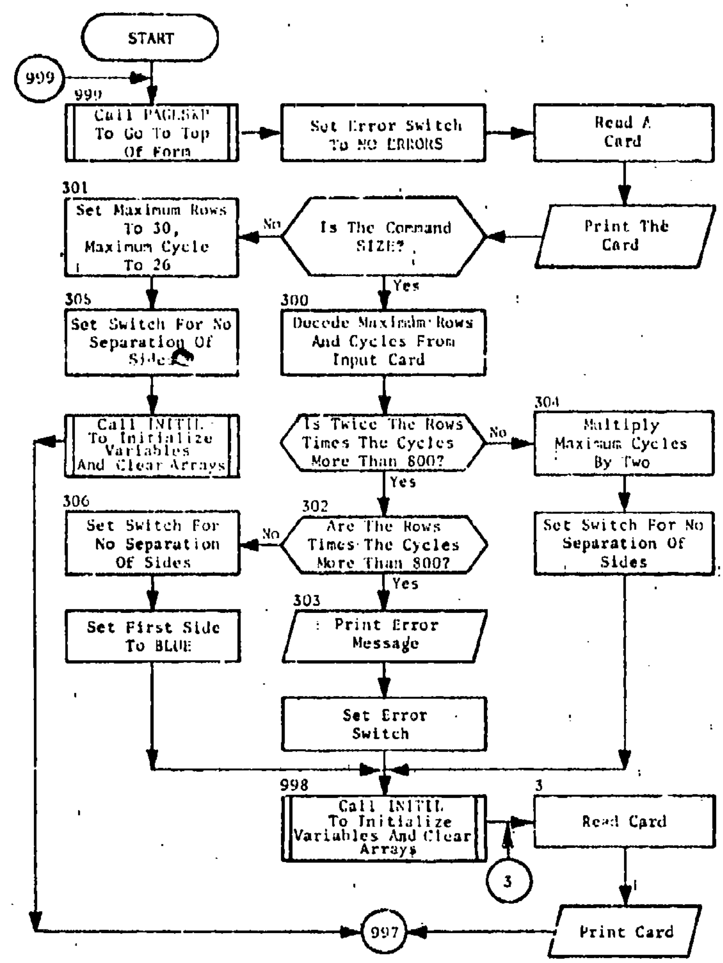


Fig. 92. Subroutine SETUP
(Sheet 1 of 6)

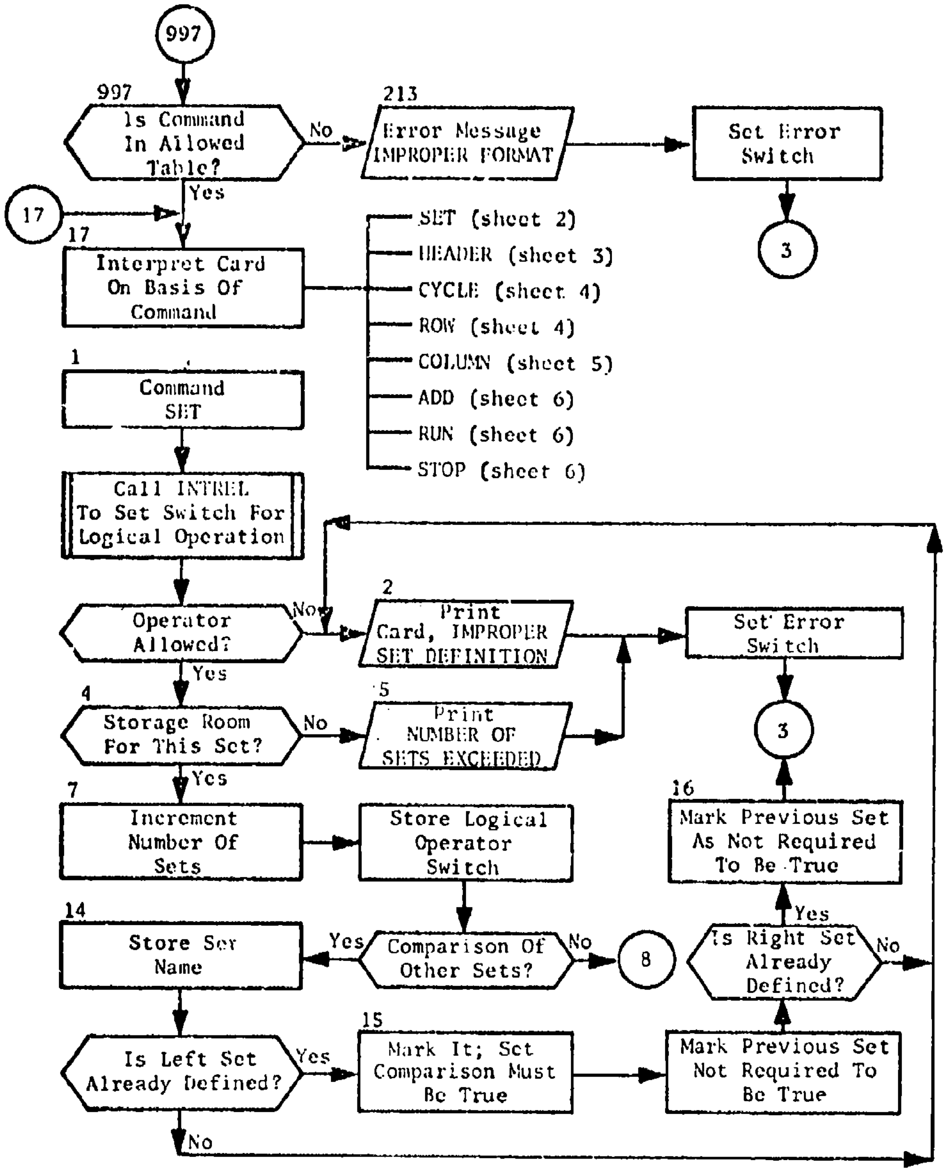


Fig. 92. (cont.)
(Sheet 2 of 6)

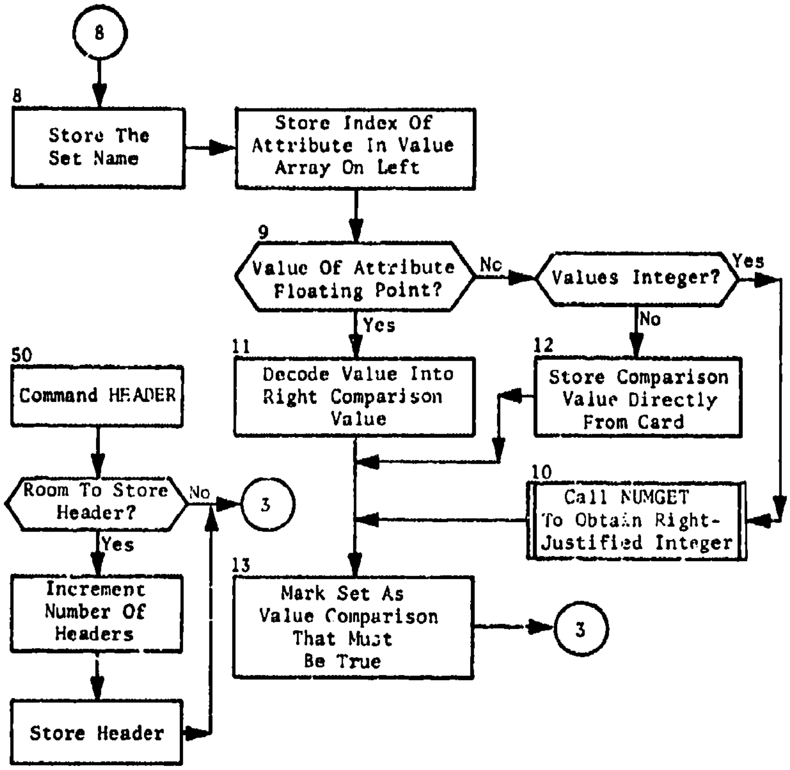


Fig. 92. (cont.)
(Sheet 3 of 6)

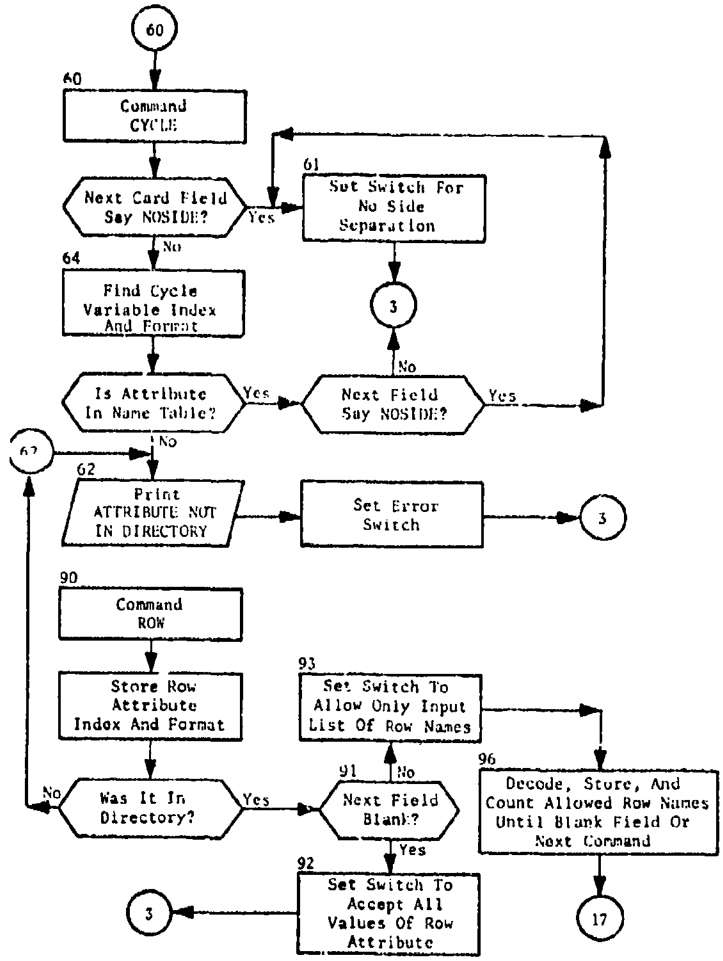


Fig. 92 (cont.)
(Sheet 4 of 6)

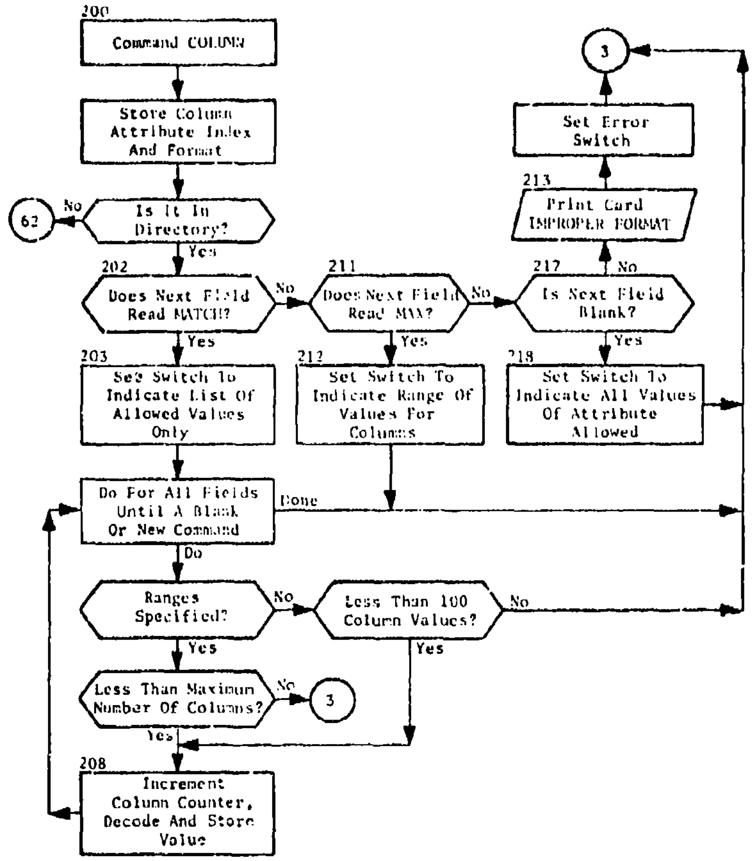


Fig. 92. (cont.)
(Sheet 5 of 6)

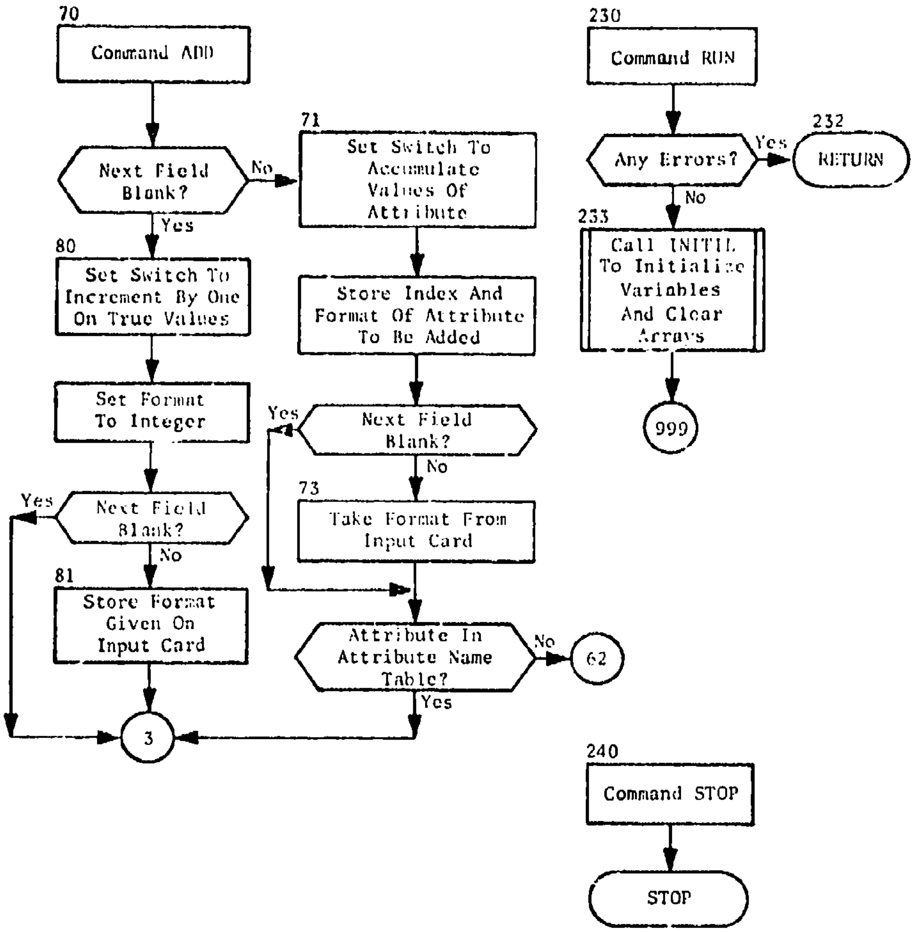


Fig. 92. (cont.)
(Sheet 6 of 6)

SUBROUTINE TABLER

PURPOSE:	To add to the count of a table entry and to accumulate the value of an attribute in the table entry.
ENTRY POINTS:	TABLER
FORMAL PARAMETERS:	None
COMMON BLOCKS:	001, ATTIND, JXCODE, PROCESS, SIZES, SWITCH, TABINDEX
SUBROUTINES CALLED:	None
CALLED BY:	TABGEN

Method

If ADDSW is two, denoting that a count of items has been specified, the table entry is incremented by one (statement 2). If ADDSW is one, calling for accumulation of the value of the attribute, the value is added to the table entry in either fixed (statement 3) or floating (statement 4) point arithmetic according to TABCODE, which was set in subroutine SETUP.

Subroutine TABLER is illustrated in figure 93.

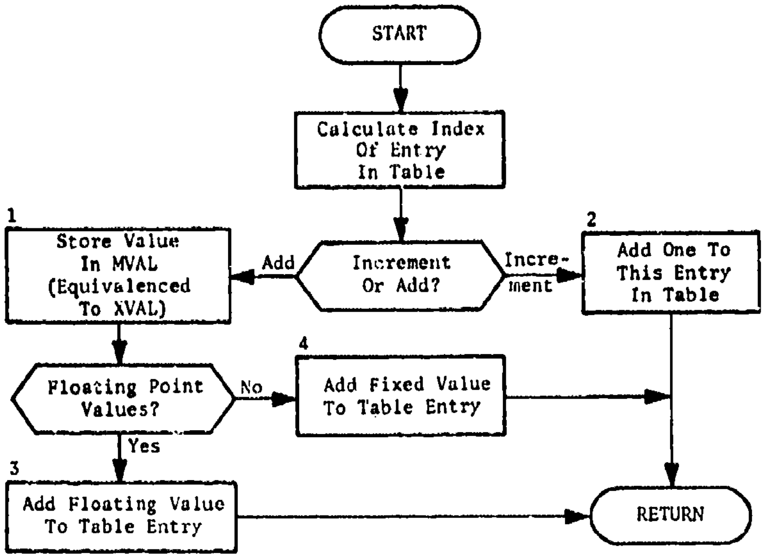


Fig. 93. Subroutine TABLER

SUBROUTINE TRUTH

PURPOSE:	To determine whether an item satisfies user-input set definitions.
ENTRY POINTS:	TRUTH
FORMAL PARAMETERS:	None
COMMON BLOCKS:	PROCESS, SETS, SWITCH
SUBROUTINES CALLED:	NVALFIND
CALLED BY:	TABGEN

Method

This subroutine determines whether an item is rejected on the basis of user-input set definitions. These have been input in this way: SETA PRIMETAR EQ 1; SETB IREG EQ 1; SETC IREG EQ 2; and SETD SETB or SETC. In this example, all Burst/Damage events in regions one and two are to be added together in the appropriate row and column of the table.

For all the required sets, TRUTH determines whether it is type 1 or 2 (statement 1) and does either type 1 (statement 5) or type 2 (statement 6) comparisons by calls on NVALFIND. If it is type 1 and is not a part of a type 2 comparison (statement 7), the item is immediately rejected if the comparison has the value false (statement 4). If it is a false type 2, it is immediately rejected. If all sets required to be true are true, the item is accepted (statement 3).

Subroutine TRUTH is illustrated in figure 94.

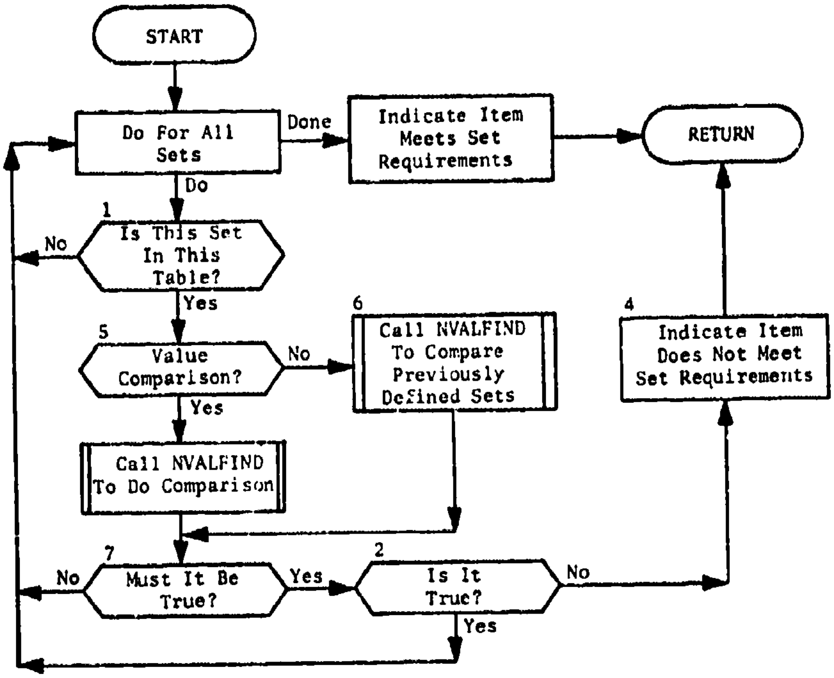


Fig. 94. Subroutine TRUTH

CHAPTER 5
PROGRAM HISTP

PURPOSE

The purpose of HISTP is to write a history of the war simulated by program SIMULATE in as much as possible the same format used by that program. This provides the user with the option of running SIMULATE with none of the detail prints and then selectively printing the detail without the cost of rerunning all of SIMULATE.

The selection criteria are the entire game history or the portion between user-input time pairs, all classes or a user-input one, all types within a class or a user-selected one, and both sides or a user-selected one.

In addition, any print command may be activated or suppressed using a data card.

INPUT FILES

The input tape is tape HISTAPE which contains the complete output of SIMULATE, the recovery tables, and the breakpoint tables.

OUTPUT FILES

There are no output files.

CONCEPT OF OPERATION

First, the print options for selection or suppression of the potential prints are read. This is followed by a selected class (or all classes), a selected type (or all types), a selected side (or both sides), and time-pairs (or entire game history). From HISTAPE, the recovery and breakpoint tables are read and stored.

Each event which has been simulated to occur by SIMULATE is now read from HISTAPE. If it meets all the user-specified criteria, the appropriate print is output according to the print options selected (table 33). In some cases, the prints will differ slightly from those in SIMULATE because of the unavailability of certain dynamic parameters. Some of the exceptions are that bomber survival of local attrition and ASM survival of local attrition cannot be differentiated, and that Enter Zone events are not recorded if penetration/depenetration is not involved.

Figure 95 illustrates the flow of program HISTP.

COMMON BLOCK DEFINITIONS

External Common Blocks

The external common blocks used by program HISTP in processing the input file are shown in table 34. Common blocks /ITP/, /TWORD/, /NOPRINT/, and /MYIDENT/ are described in the Programming Specifications Manual, Volume I, Part A, Appendix A.

Internal Common Blocks

There are no internal common blocks used by program HISTP.

Table 33. Prints Controlled by JPRINT
(Sheet 1 of 2)

EVENT	PRINT OPTION DESCRIPTION
1 Missile Launch	1 Missile Launch
	2 Results of attempted missile launch
	3 All launched, or all targets covered
	4 Launches
2 Bomber Launch	6 Successful bomber or tanker launch
	7 Delay
	8 Dead aircraft launch base
3 Complete Launch	9 Killed during powered flight
	47 MIRV success
	48 MIRV target list
4 Refuel	10 Successful refueling
	11 Refuel abort
	12 No tanker available
	13 Return home
5 Enter Zone	15 Enter enemy territory
	16 Leave enemy territory
6 Zone Status	17 Status of defense zones
7 Area Attrition	18 Area attrition
8 Local Attrition	19 Bomber survives
	20 Bomber kill (before/after)
	21 ASM survives
	22 ASM kill
	23 Dud warhead
9 Terminal BMD	24 Terminal BMD results
10 Burst/Damage	25 Warhead, coordinate data
	26 Result of detonation
	27 Collocation offset distances
11 Enter Refuel Area	28 Tanker arrival at refuel area

Table 33. (cont.)
(Sheet 2 of 2)

EVENT	PRINT OPTION DESCRIPTION
12 Leave Refuel Area	29 Tanker leave refuel area empty
	30 Tanker leave refuel area with fuel
	31 Tanker abort while on station
13 Abort	32 Scheduled abort
	33 Random abort
14 ASM Launch	34 Successful ASM launch
	35 ASM launch failure
15 Decoy Launch	36 Decoy launch
16 Recover	37 Successful recovery
	38 Failure -- base dead on arrival
	43 No base on depenetration
	44 Recovery base saturated
	50 Recover at home base
	51 Home base dead on arrival
17 Change Altitude	39 Change altitude
18 Area BMD	40 Selection for defense
	41 Allocation of area BMD
	42 Random area BMD
19 Recheck	45 Base killed after recovery
	49 Home base killed after recovery
	46 Recovery data
20 Determine time of naval attrition	52 Scheduled for naval attrition
21 Naval attrition	53 Naval attrition

Table 34. Program HISTP External Common Blocks

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
BRKPNT	NTYPECUM(15)	Cumulative number of types through end of class
	NBLUETYP(15)	Number of BLUE types in class
	INDBEGCL(15)	Beginning index of class
	INDBEGTY(250)	Beginning index of type
NAMES	NAMESIDE(2)	Name of side
	NAMECLAS(15)	Name of class
	NAMETYPE(250)	Name of type
HISTOUT	NHISTOUT(200)	Simulator event data; see program READSUM for description

* Parenthetical values indicate array dimensions. All other elements are single word variables.

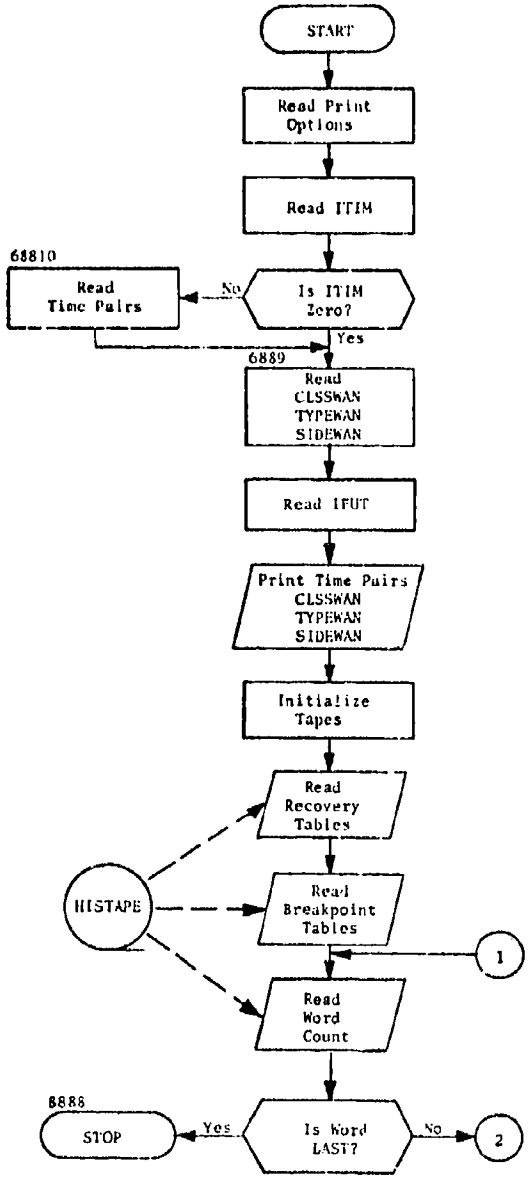


Fig. 95. Program HISTP
(Sheet 1 of 11)

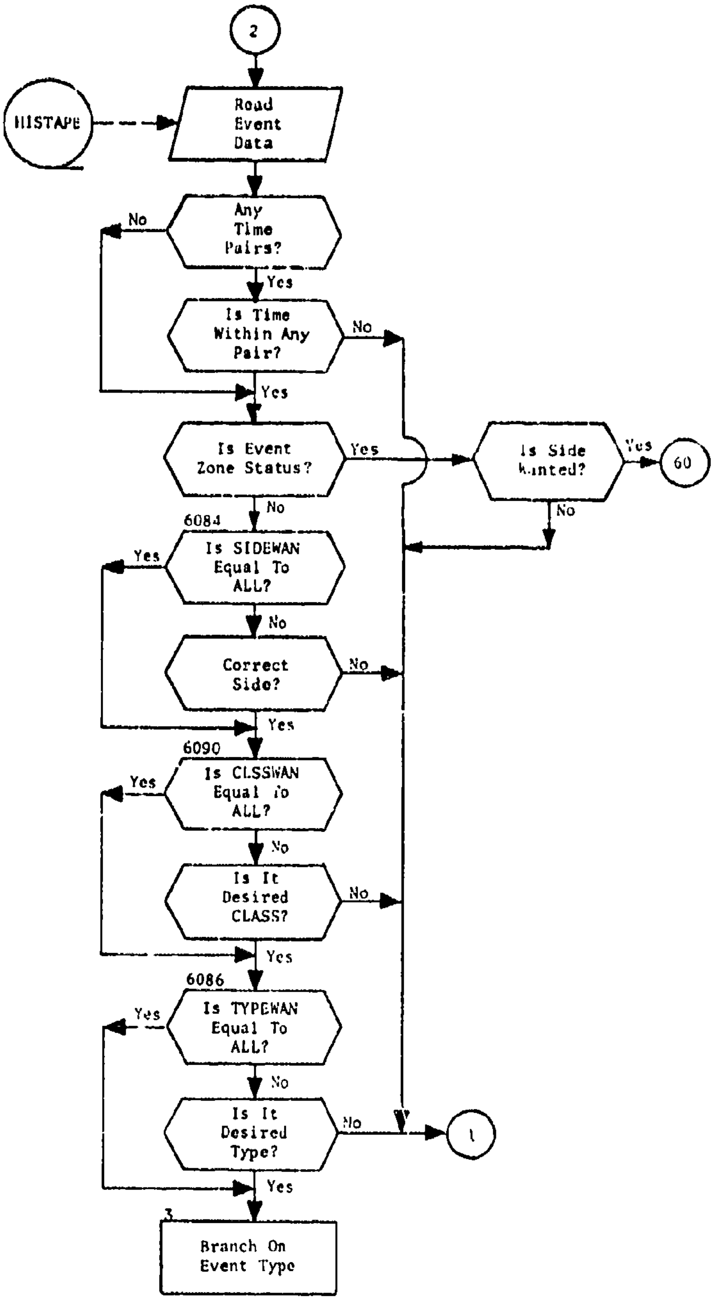


Fig. 95. (cont.)
(Sheet 2 of 11)

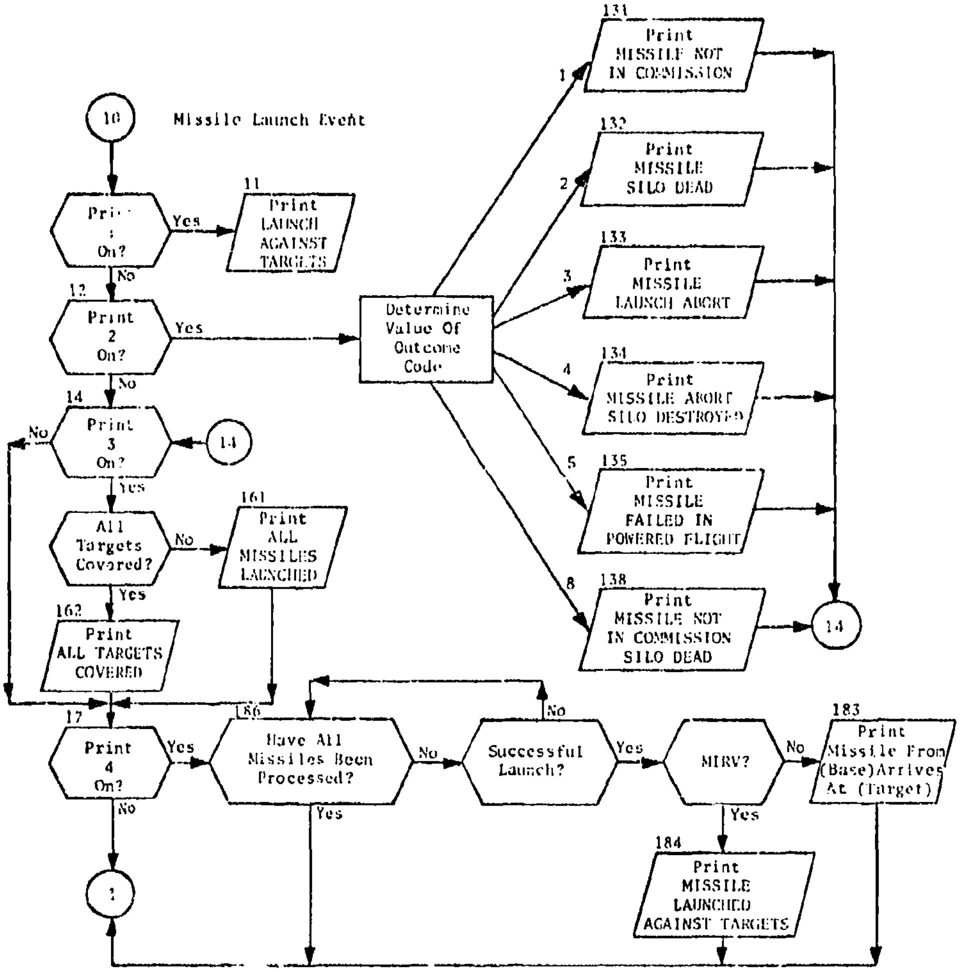
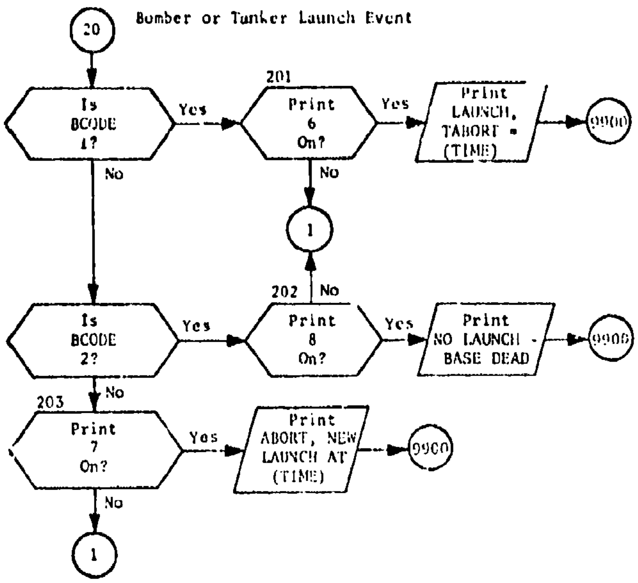


Fig. 95. (cont.)
(Sheet 3 of 11)



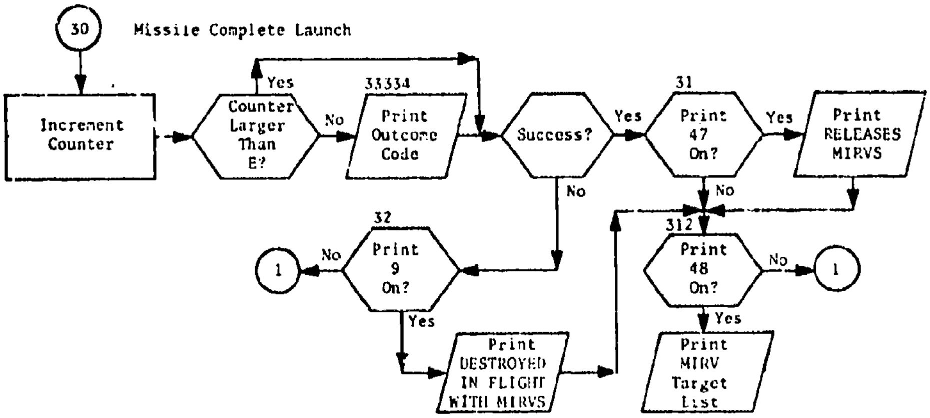
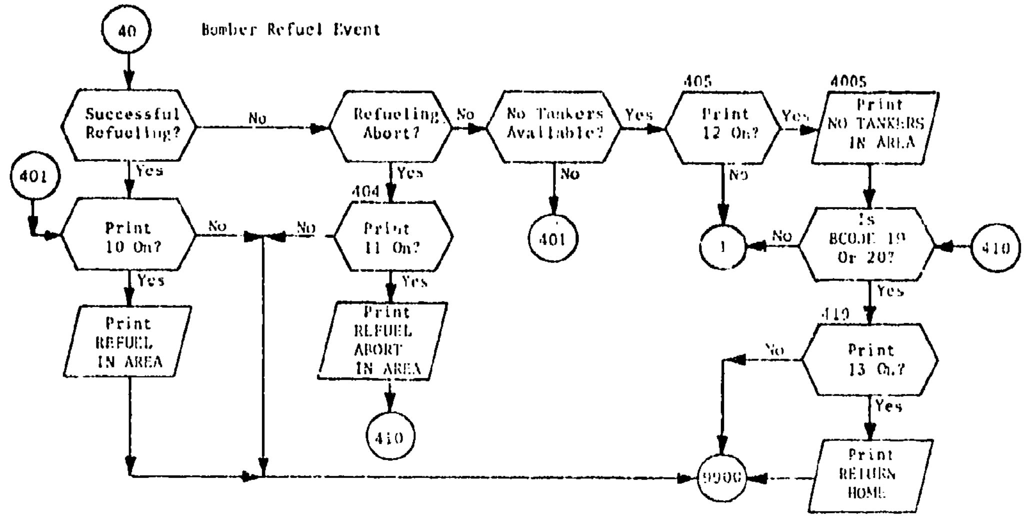


Fig. 95. (cont.)
(Sheet 4 of 11)



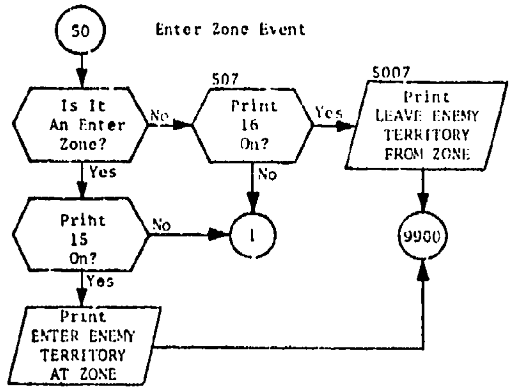


Fig. 95. (cont.)
(Sheet 5 of 11)

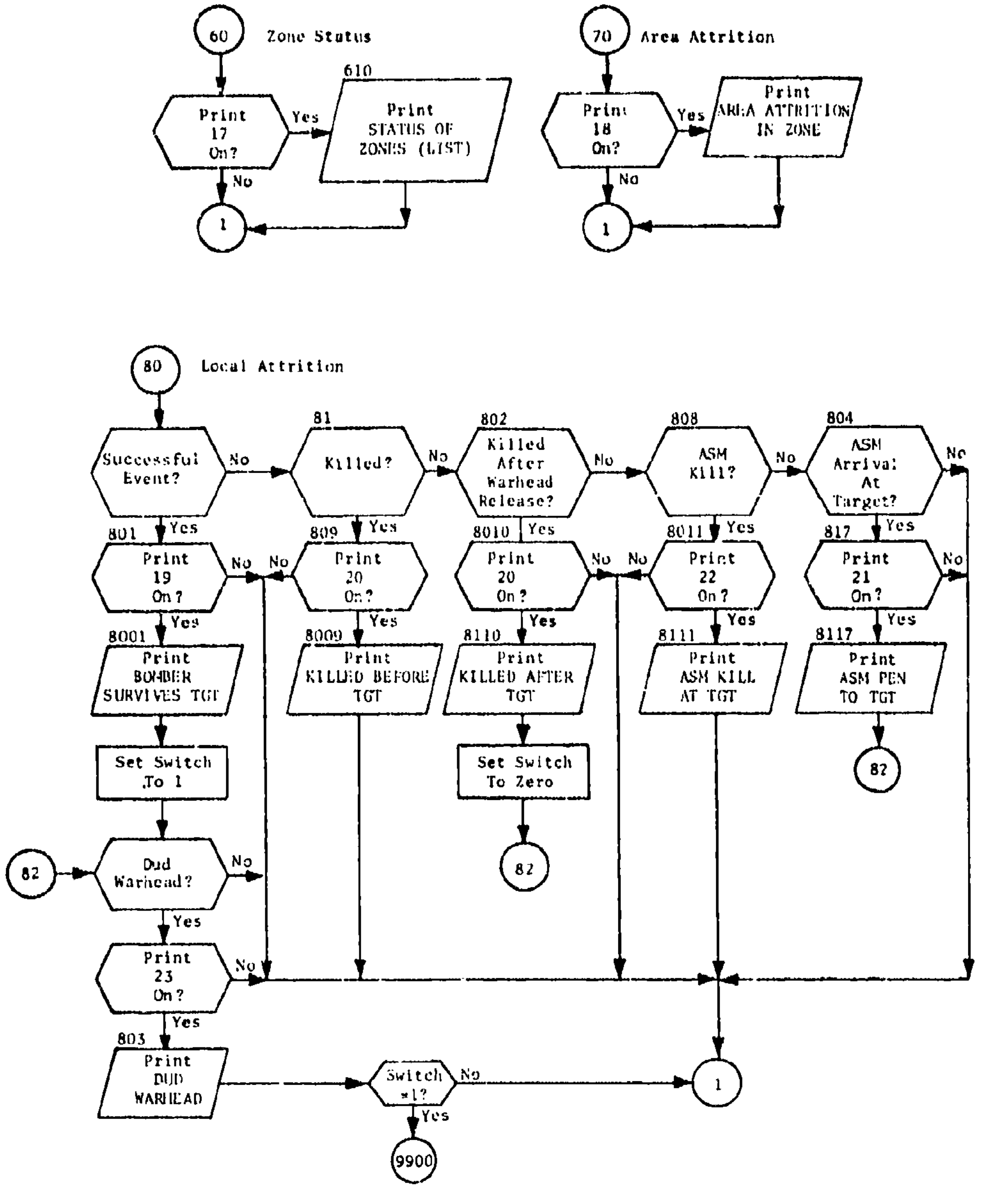


Fig. 95. (cont.)
(Sheet 6 of 11)

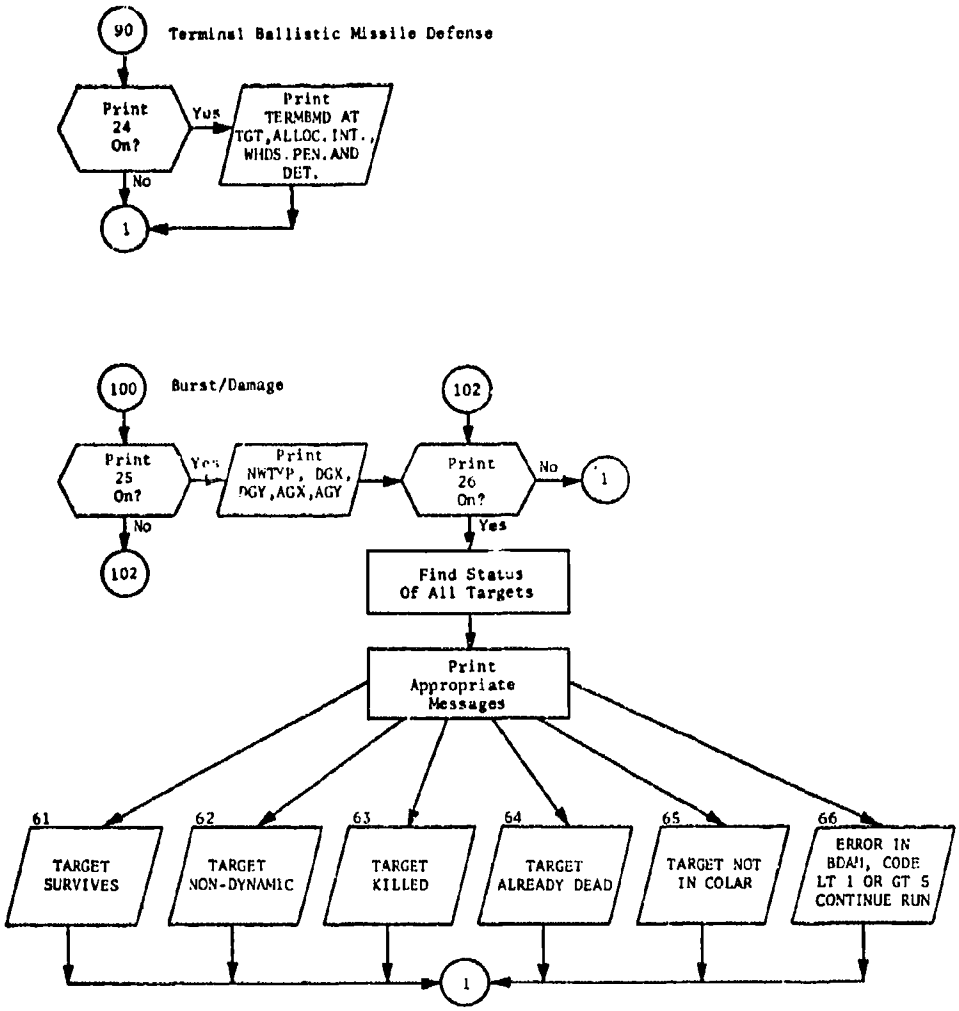


Fig. 95. (cont.)
(Sheet 7 of 11)

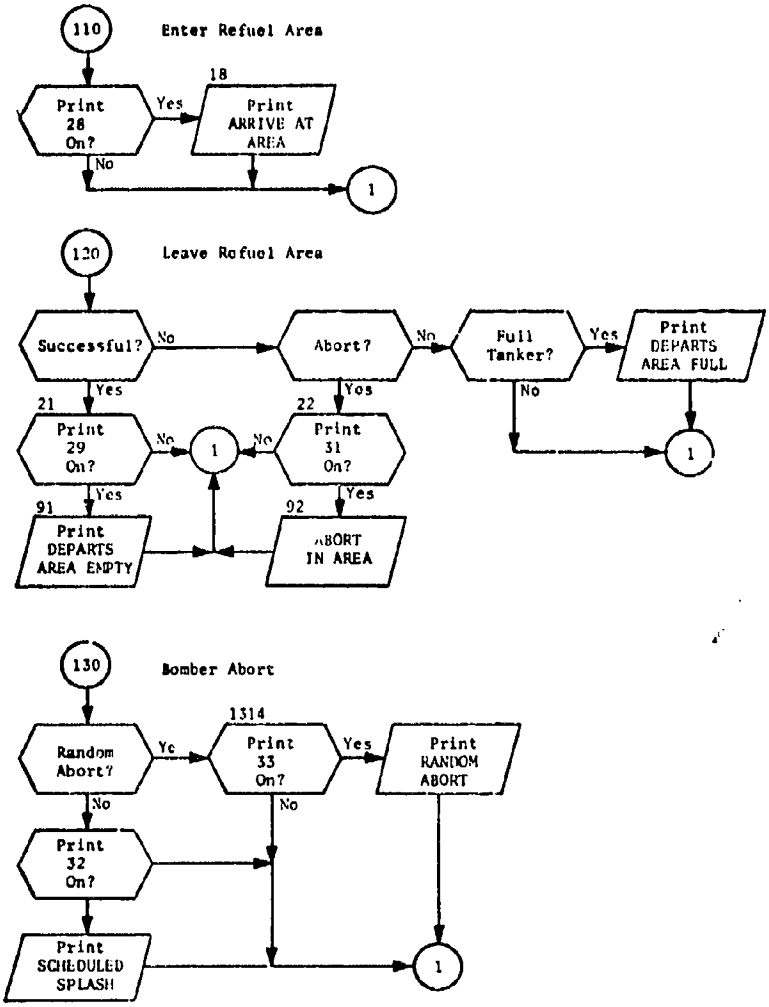


Fig. 95. (cont.)
(Sheet 8 of 11)

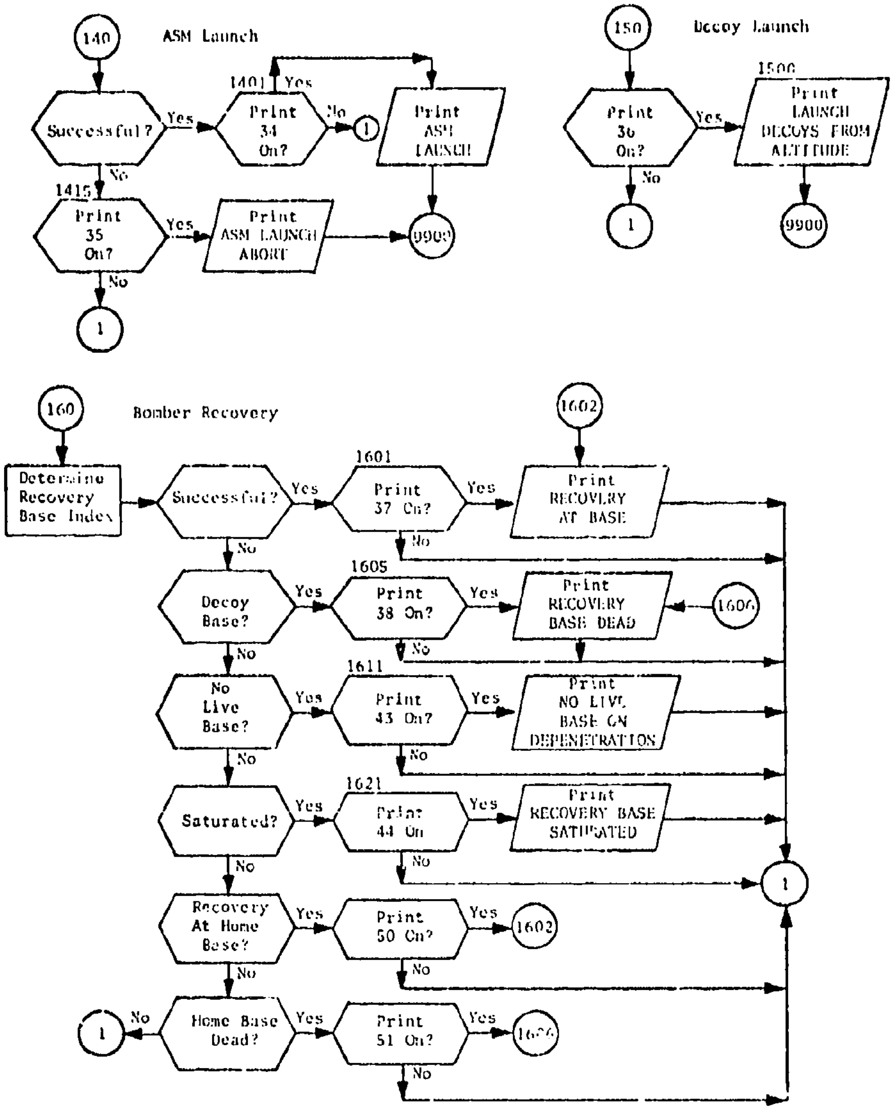


Fig. 95. (cont.)
(Sheet 9 of 11)

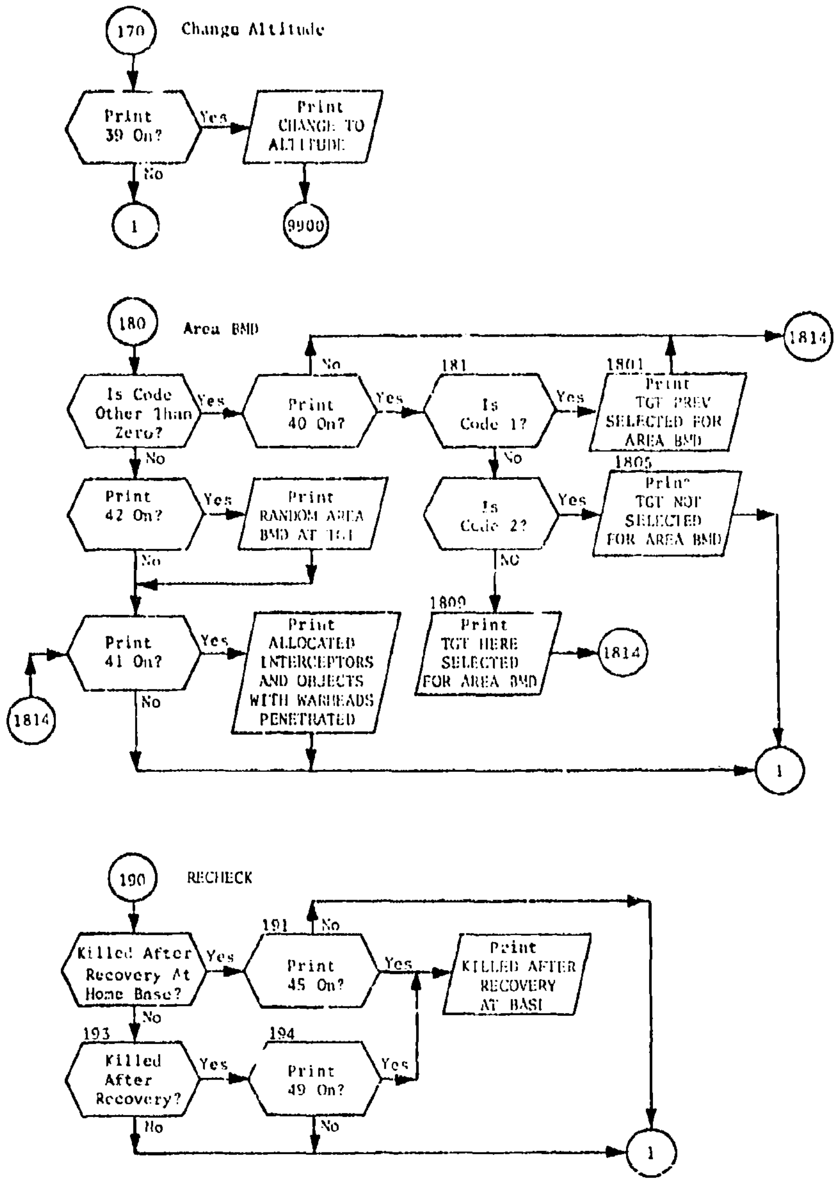


Fig. 95. (cont.)
(Sheet 10 of 11)

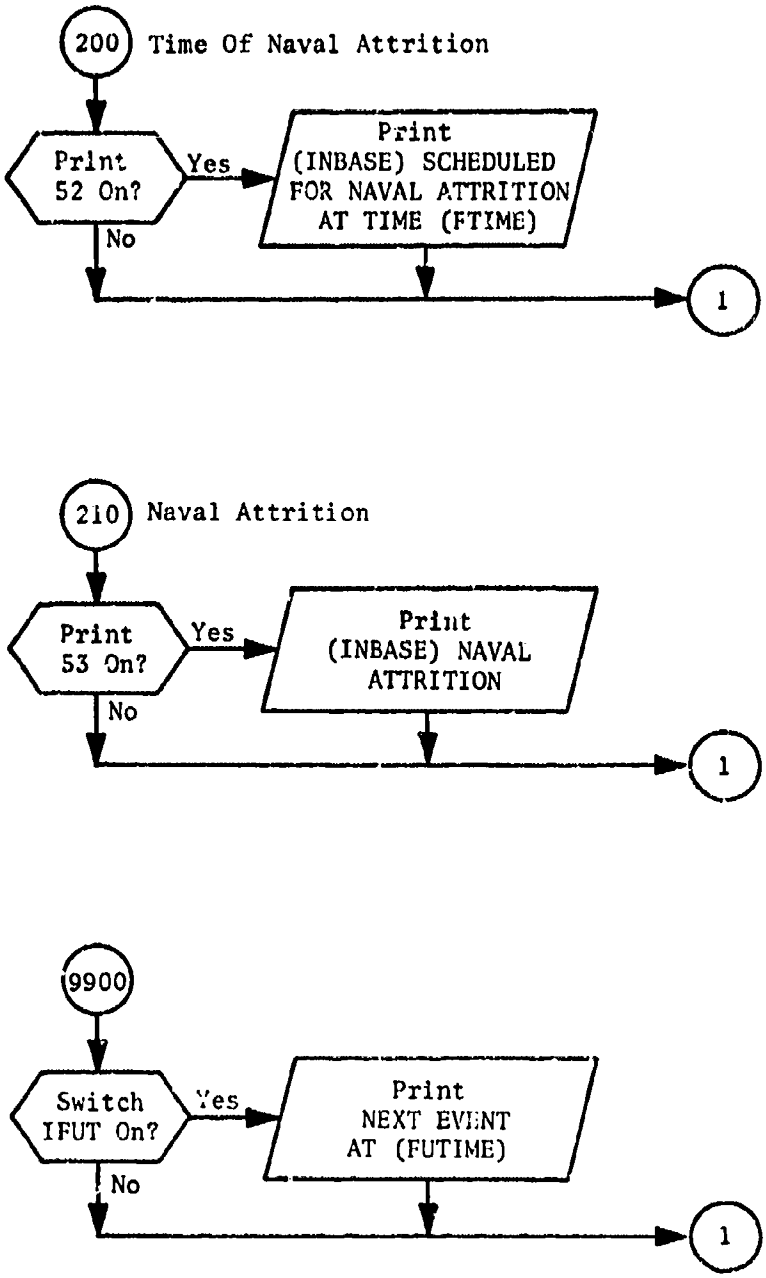


Fig. 95. (cont.)
(Sheet 11 of 11)

APPENDIX A
QUICK ATTRIBUTE NAMES AND DESCRIPTIONS

ATTRIBUTE NAME	DESCRIPTION
ABRATE	Probability of aircraft in-flight abort per hour of flying time
ADBLI	ALERTDBL probability for initiative attack
ADBLR	ALERTDBL probability for a retaliatory attack
ADEFCMP	Area ballistic missile defense (BMD) component index (radar or missile launch site)
ADEFZON	Area ballistic missile defense (BMD) zone number
AGX	Offset X-coordinate of AGZ (fiftieths of nautical miles)
AGY	Offset Y-coordinate of AGZ (fiftieths of nautical miles)
AHOB	Actual height of burst of weapon (air or ground)
ALERTDBL	Probability of destruction before launch (DBL) of alert delivery vehicle (missile or bomber)
ALERTDLY	Delay of alert vehicle before commencing launch (hours)
AREA	Area of a bomber defense ZONE (millions of nautical miles^2)
ASMTYPE	Air-to-surface missile type
ATTRCORR	Attrition parameter for a bomber corridor (probability of attrition per nautical mile)
ATTRLEG	Attrition parameter for each route leg in bomber sortie (probability of attrition per nautical mile)
ATTRSUPF	Amount of original attrition that remains after defense suppression

ATTRIBUTE NAME	DESCRIPTION
AZON1	First area defense zone covered by a BMD long-range radar
AZON2	Second area defense zone covered by a BMD long-range radar
AZON3	Third area defense zone covered by a BMD long-range radar
BCODE	Code indicating the outcome of a simulated bomber event
BENO	Bombing encyclopedia number
BLEGNO	Index to boundary line segment
CATCODE	Category Code as reflected in Joint Resource Assessment Data Base (JAD)
CCREL	Regional reliability of offensive command and control (probability)
CEP	Circular error probable (CEP), delivery error applicable to bomber and missile weapons (nautical miles)
CLASS	Class name assigned to identify sets of TYPES in data base
CLASST	Target CLASS
CNTRYLOC	Country code for country where item is located
CNTRYOWN	Country code for country which owns the item
CNTYLOCT	Target country code for country where the target is located
CNTYOWNT	Target country code for country which owns the target
CODE	Outcome code for a general event used in simulation
CPACTY	Capacity of a bomber recovery base (number of vehicles)

ATTRIBUTE NAME	DESCRIPTION
DATEIN	Earliest date in inventory (year)
DATEOUT	Latest date in inventory (year)
DEFRANGE	Typical range of interceptors at defense bases near a corridor (nautical miles)
DELAY	Delay time (e.g., launch delay time) (hours)
DELTA	Time interval between successive vehicle launches from the same base (missile or bomber) (hours)
DESIG	Target designator code, e.g., AB100, which uniquely identifies each target element included in the data base
DGX	Offset X-coordinate of desired ground zero (DGZ) (fiftieths of nautical miles)
DGY	Offset Y-coordinate of DGZ (fiftieths of nautical miles)
DHOB	Height of burst of weapon (0-ground, 1-air)
EFECNES1, EFECNES2	Attributes assigned to fighter interceptor units (ICLASS = 5 in the data base): the value EFECNES1 or EFECNES2 is assigned to the attribute EFECTNES depending on value of BASEMOD input parameter POSTURE (if POSTURE=1, EFECNES1 is used, otherwise EFECNES2 value is assigned)
EFECTNES	Air defense capability (arbitrary scale) established by user to indicate relative effectiveness of air defense command and control installations and fighter interceptor bases
EVENT	Index to event type
EVENTN	Index to type of event which did not occur
FFRAC	Fission fraction (fission yield/total yield)
FLAG	Numeric code (1 through 9 permitted) used to impose restrictions on the allocation of weapons within QUICK

ATTRIBUTE NAME	DESCRIPTION
FLTNO	Flight number for a sortie
FUNCTION	Operational application code for a weapon system (e.g., ICBM)
FVALH1	Fraction of value of target in first hardness component
FVALT1	Franction of target value that disappears by T1 (percent)
FVALT2	Franction of target value that disappears by T2 (percent)
H1	First hardness component of a target (VULN)
H2	Second hardness component of a target (VULN)
HILOATTR	The ratio of the low-altitude attrition rate to the high-altitude rate (decimal fraction)
IALERT	Alert status; 1= alert, 2 = nonalert
IALT	Altitude index (1 = high, 0 = low)
IATTACK	Selection index for preferential area BMD; 1 forces target selection for defense.
ICLASS	Class index assigned for game
ICLASST	Target class index
ICOMPLEX	Complex index
ICORR	Bomber corridor index number assigned in program PLANSET: 1 - Tactical (FUNCTION=TAC) aircraft corridor (TYPE name DUMMY in the data base) 2 - Naval attack corridor (TYPE name NAVALAIR in the data base) used by bomber units with PKNAV greater than zero

ATTRIBUTE NAME	DESCRIPTION
ICORR (cont.)	>2 - Other corridors used by long-range bombers (FUNCTION=LRA)
IDBL	Index to data tables for time-dependent destruction before launch probability
IDUD	Dud warhead indicator; assigned to weapons which arrive at the target but fail to detonate; 1=dud warhead
IGIW	Indices of General Industrial Worth (IGIW) (dollars)
IGROUP	Group index assigned for weapon grouping during game
IMIRV	Identifying index for system with multiple independently targetable re-entry vehicles
INDEXNO	Index of a data base item (potential target) used during processing to identify the item
INDV	Vehicle index within base
INTAR	Target index (corresponds to INDEXNO)
IPENMODE	Penetration mode; 1 = aircraft uses penetration corridor, 0 = penetration corridor not used
IPOINT	Index to a geographic point
IRECMODE	Recovery mode; 1 = aircraft should plan recovery, 0 = aircraft recovery not planned
IREFUEL	Bomber refueling code
IREG	Index to identify a geographic region
IREP	Reprogramming index (capability of missile squadron)
ISITE	Site number
ITGT	Target index number assigned by Plan Generation subsystem

ATTRIBUTE NAME	DESCRIPTION
ITIME	Index to time periods in time-dependent DBL data tables
ITYPE	Type index assigned for game
ITYPET	Target type index
IVULN	Index to vulnerability number table
IWTYP2	Second warhead type
JTYPE	Type index within class
JTYPET	Target type index within class
KORSTYLE	Parameter to adjust mode of corridor penetration
LAT	Latitude (degrees)*
LEGNO	Index to line segment
LINK	The index of a leg linked to the current point
LONG	Longitude (degrees)*
MAJOR	Major reference number as reflected in the Joint Resource Assessment Data Base (JAD)
MAXFRACV	Maximum value of weapon resources to be used relative to target value (in processing MAXCOST=MAXFRACV)
MAXKILL	Desired maximum damage expected for a target

* Latitude and longitude are carried internally in the QUICK system in the following format:

North latitude	0. (equator) to +90. (North Pole)
South latitude	0. (equator) to -90. (South Pole)
East longitude	180. to 360. (Greenwich Meridian)
West longitude	0. (Greenwich Meridian) to 180.

These attributes may be input in either the above format or in standard degree, minute, second, direction format.

ATTRIBUTE NAME	DESCRIPTION
MINKILL	The required minimum damage established for a target
MINOR	Minor reference number as reflected in JAD to identify an item
MISDEF	Number of terminal ballistic missile interceptors for a target
MVA	Manufacturing value added (MVA); indicates the amount of value added by manufacture within a specific area (expressed in U. S. dollars)
MWHDS	Number of missile warheads penetrating area defenses to terminal defense
NADBLI	NALRTDBL for initiative attack
NADBLR	NALRTDBL for retaliatory attack
NAINT	Number of area ballistic missile interceptors at an interceptor launch base
NALRTDBL	Probability of destruction before launch (DBL) of non-alert vehicle
NALRTDLY	Delay of non-alert vehicle before commencing launch (hours)
NAME	Arbitrary alphameric descriptor for any item included in the data base
NAREADEC	Number of decoys per independent re-entry vehicle for area BMD
NASMS	Number of ASMs carried by a bomber
NCM	Number of countermeasures carried by vehicle
NDECOYS	Number of decoys on a bomber or number of decoys per independent re-entry vehicle for terminal BMD
NDET	Number of warheads detonating in current event
NEXTZONE	The adjacent zone to a side of a defense zone
NMPSITE	Number of missiles per site

ATTRIBUTE NAME	DESCRIPTION
NOALERT	Number of vehicles on alert at a base
NOBOMB1	Number of first bomb type carried by vehicle
NOBOMB2	Number of second bomb type carried by vehicle
NOINCOM	Number of delivery vehicles in commission
NOPERSQN	Number of weapon vehicles per squadron
NOPERSQ1 NOPERSQ2 NOPERSQ3	Attributes used in program BASEMOD to compute the value of the attribute NOPERSQN for bomber units; numbers 1, 2, and 3 specify surprise, initiative, and retaliatory attack plans, respectively
NPEN	Number of warheads penetrating in current event
NTARG	Number of targets in missile launch event
NTINT	Number of terminal BMD interceptors at target
NWHDS	Number of warheads per independent re-entry vehicle (missiles)
NWPNS	Number of weapons in a group
NWTYPE	Warhead type
PARRIVE	Probability of bomber arrival in current event
PAYLOAD	Index which identifies entire weapon and penetration aid complement on a vehicle
PDES	Probability that launch failure destroys missile
PDUD	Probability a warhead will fail to detonate
PEN	Penetration probability for a weapon
PFPF	Probability of failure during powered flight (missiles)

ATTRIBUTE NAME	DESCRIPTION
PINC	Probability that a missile is in commission
PKMIS	Probability a missile fails to penetrate terminal defense
PKNAV	Single shot kill probability of a weapon against a naval target (a value greater than zero restricts weapon use to naval targets)
PLABT	Probability of vehicle launch abort
PLACE	Index to geographic location of an event
PLACEN	Index to geographic location of an event which did not occur
POP	Population (cities) (thousands)
POSTURE	Force readiness condition
PRABT	Probability of refueling abort
PRIMETAR	Prime target flag; 1 signifies priority target in a complex
PSASW	Destruction before launch probability assigned a weapon for a specified time period
RADIUS	Size descriptor for area targets (nautical miles)
RANGE	Vehicle range (nautical miles)
RANGEDEC	Range decrement for low-altitude aircraft flight (high range/low range)
RANGEREF	Range (nautical miles) of bomber with refueling
REL	Reliability - probability that weapon system will arrive at target given successful launch
RESERVE	Technique used to remove certain targets from weapon allocation when RESERVE = 0
SIDE	Item side name, currently either "RED" or "BLUE"
SITENO	Site number (currently for individual missile sites)

ATTRIBUTE NAME	DESCRIPTION
SPDLO	Speed at low altitude (knots)
SPEED	Speed (knots)
SQNNO	Squadron number
T1	Time of departure of first value component of a target
T2	Time of departure of second value component of a target
T3	Time of departure of third value component of a target
TAIM	Number of aim points perceived by terminal defense in current event
TARDEFHI	Level of local bomber defense at high altitude*
TARDEFLO	Level of local bomber defense at low altitude*
TASK	Target task code indicating targeting priority
TGTSTAT	Indicates target status as dynamic or nondynamic; in simulation status (alive/dead) is maintained for dynamic targets
TIME	Game time at which event occurred (hours)
TIMEN	Time planned for event which did not occur (hours)
TMDEL	Mean delay time to relaunch after a nondestructive aircraft abort (hours)
TPASW	Time at which a time period ends for DBL data tables; there may be up to 10 time periods for each table
TRETARG	Time required to retarget for known in-flight missile aborts (hours)

---

* Arbitrary units scaled by user-input parameter in Plan Generation subsystem. Minimum value 0 for no defense. Highest allowed defense level is +7.

ATTRIBUTE NAME	DESCRIPTION
TTOS	Total time on station (for a tanker) (hours)
TVUL	Time a missile remains within vulnerable range of launch site (hours)
TYPE	Arbitrary alphameric designator (type name) to identify smallest sets in data base
TYPET	Target TYPE
TYPE1 } TYPE2 }	Attributes assigned fighter interceptor units (ICLASS=5 in the data base): attribute TYPE is assigned the TYPE1 or TYPE2 value based on BASEMOD input parameter POSTURE (POSTURE=1 TYPE1 is used; otherwise TYPE2 value used)
VAL	Relative value of an item within its CLASS as established in the data base by the user
VALU	Game value of an item (assigned in plan generation based on user-input parameters)
VAL1 } VAL2 }	Attributes assigned fighter interceptor units (ICLASS=5 in the data base): attribute VAL is assigned the VAL1 or VAL2 value based on BASEMOD input parameter POSTURE (POSTURE=1, VAL1 is used; otherwise VAL2 value is assigned)
VULN	Vulnerability number
WACNO	World aeronautical chart number
WHDTYPE	Warhead type index assigned in the data base
WHDTYPEN	Warhead type index (used with EVENTN)
YIELD	Yield (MT)
ZONE	An area bomber defense zone enclosed by a set of linked boundary points

APPENDIX B
ENTRY POINTS FOR QUICK UTILITY ROUTINES

This appendix contains an alphabetic listing of the entry points associated with all utility programs and subroutines. Subroutines associated with each of these entry points are indicated below.

ENTRY POINT	TO SUBROUTINE
ABORT	ABORT
ALOCDIR	FILEHNR
ANOTHER	ANOTHER
ATN2PI	ATN2PI
CHANGE	CHANGE
CLOSPIL	CLOSPIL
CLRCMON	CLRCMON
DEACTIV	FILEHNR
DECLARES	DECLARES
DELLONG	DELLONG
DIFFLNG	DIFFLONG
DIFFLONG	DIFFLONG
DISTF	DISTF
DSTF	DSTF
ENDDATA	ENDDATA
ENDTAPE	ENDTAPE
ERAZE	ERAZE
EQUIV	EQUIV
FILEDUMP	FILEDUMP
FILEHNR	FILEHNR
GETCLK	GETCLOCK

ENTRY POINT	TO SUBROUTINE
GETCLOCK	GETCLOCK
GETDATE	GETDATE
GETDF	GETDF
GETLIMIT	GETLIMIT
GETLOC	GETLOC
GETVALU	GETVALU
IGET	IGET
INBUFDK	INBUFDK
INERRDK	INERRDK
INERRTP	INERRTP
INITAP	FILEHNR
INITAPE	FILEHNR
INITEDIT	INITEDIT
INITEDT	INITEDIT
INLABEL	INLABEL
INPITEM	INPITEM
INTERP	INTERP
INTERPGC	INTERPGC
INTRPGC	INTERPGC
IPUT	IPUT
ITLE	ITLE
IWANT	IWANT
KEYMAKE	KEYMAKE
LOCF	LOCF
LOCREAD	LOCREAD
LOCWRIT	LOCREAD
LOCWRITE	LOCREAD
NEWUNIT	NEWUNIT
NEXTAPE	NEXTAPE
NEXTFILE	NEXTFILE

ENTRY POINT	TO SUBROUTINE
NEXTITEM	INITITEM
NEXTITM	INITITEM
NODIRC	NODIRC
NUMGET	NUMGET
OPENSPL	OPENSPL
ORDER	ORDER
OUTBFDK	OUTBFDK
OUTBFTP	OUTBFTP
OUTDF	OUTDF
OUTERDK	OUTERDK
OUTERTP	OUTERTP
OUTFILE	OUTFILE
OUTITEM	OUTITEM
OUTWORDS	OUTWORDS
OUTWRDS	OUTWORDS
PAGESKIP	PAGESKP
PAGESKP	PAGESKP
PRITEM	PRITEM
PRNTBAS	PRNTBASE
PRNTBASE	PRNTBASE
PRNTBSE	PRNTBASE
PRNTDATA	PRNTDTA
PRNTDTA	PRNTDTA
PRNTDIRC	PRNTDIRC
PRNTDRC	PRNTDIRC
PRNTLAB	FILEHNR
PRNTPAGE	PRNTPGE
PRNTPGE	PRNTPGE
RDARRAY	RDARRAY
READDIR	READDIR
RELOADF	RELOADF

ENTRY POINT	TO SUBROUTINE
REORDER	REORDER
SETHEAD	SETHEAD
SETREAD	SETREAD
SETWRIT	FILEHNR
SETWRITE	FILEHNR
SKIP	SKIP
SSKPC	SSKPC
STORAGE	STORAGE
TERMTAP	TERMTAP
TERMTAPE	TERMTAP
TERMTPE	TERMTAP
TIMEDAY	TIMEDAY
TIMEME	TIMEME
WARNING	ABORT
WRARRAY	RDARRAY
WRITEDIR	WRITEDIR
WRITEDR	WRITEDIR
WRWORD	FILEHNR

DEFENSE COMMUNICATIONS AGENCY
NATIONAL MILITARY COMMAND SYSTEM
SUPPORT CENTER
WASHINGTON, D.C. 20301

IN REPLY REFER TO B221

13 October 1972

AD-742 790

TO: DISTRIBUTION

SUBJECT: Change 1 to Programming Specifications Manual CSM PSM 9A-67, Volume III, Simulation Output Subsystem, Part A.

1. The purpose of this set of change pages is to document changes necessary to correct an error in the Quick-Reacting General War Gaming System (QUICK) where user control of simulator prints 7 (Bomber Launch Delays) and 19 (Bomber Survives Local Attrition) inadvertently affected the simulation logic. These change pages should be inserted in the manual to accurately reflect the currently operational version of QUICK at the National Military Command System Support Center.

2. A list of Effective Pages to verify the accuracy of the manual is enclosed. This list should be inserted before the title page and an appropriate entry made in the Record of Changes for the manual.

FOR THE COMMANDER:

12 Enclosures
Change 1 pages

J. DOUGLAS POTTER
Chief, Military Personnel and
Administrative Services Office

CH-1

EFFECTIVE PAGES - 25 August 1972

This list is used to verify the accuracy of CSM PSM 9A-67, Volume III, Part A, after change 1 pages have been inserted. Original pages are indicated by the letter O, and change 1 by the numeral 1.

Page No.	Change No.
Title Page, Part A	0
ii-xii, Part A	0
1-113	0
114	1
115-181	0
182	1
182.1-182.2	1
183	1
184-355	0
356	1
357-358	0
Title Page, Part B	0
ii-vii, Part B	0
359-865	0

DISTRIBUTION

Addressee	Copies
NMCSSC Codes	
B121	3
B122 (stock)	6
B200	1
B210	2
B220	29
B600	1
DCA Codes	
920	1
950	1
System Engineering Facility, ATTN: T221 Reston, Virginia 22070	1
OJCS Studies, Analysis and Gaming Agency, ATTN: SFD, Room 1D957, Pentagon, Washington, D.C. 20301	5
Commander-in-Chief, North American Air Defense Command ATTN: NPPG, Ent Air Force Base, Colorado 80912	2
Commander, U.S. Air Force Weapon Laboratory (AFSC) ATTN: AWL, Kirtland Air Force Base, New Mexico 87117	2
Director, Strategic Target Planning Offutt Air Force Base, Nebraska 68113	2
Chief of Naval Operations, ATTN: OP963G Room 5E531, Pentagon, Washington, D.C. 20350	2
Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314	12
	70

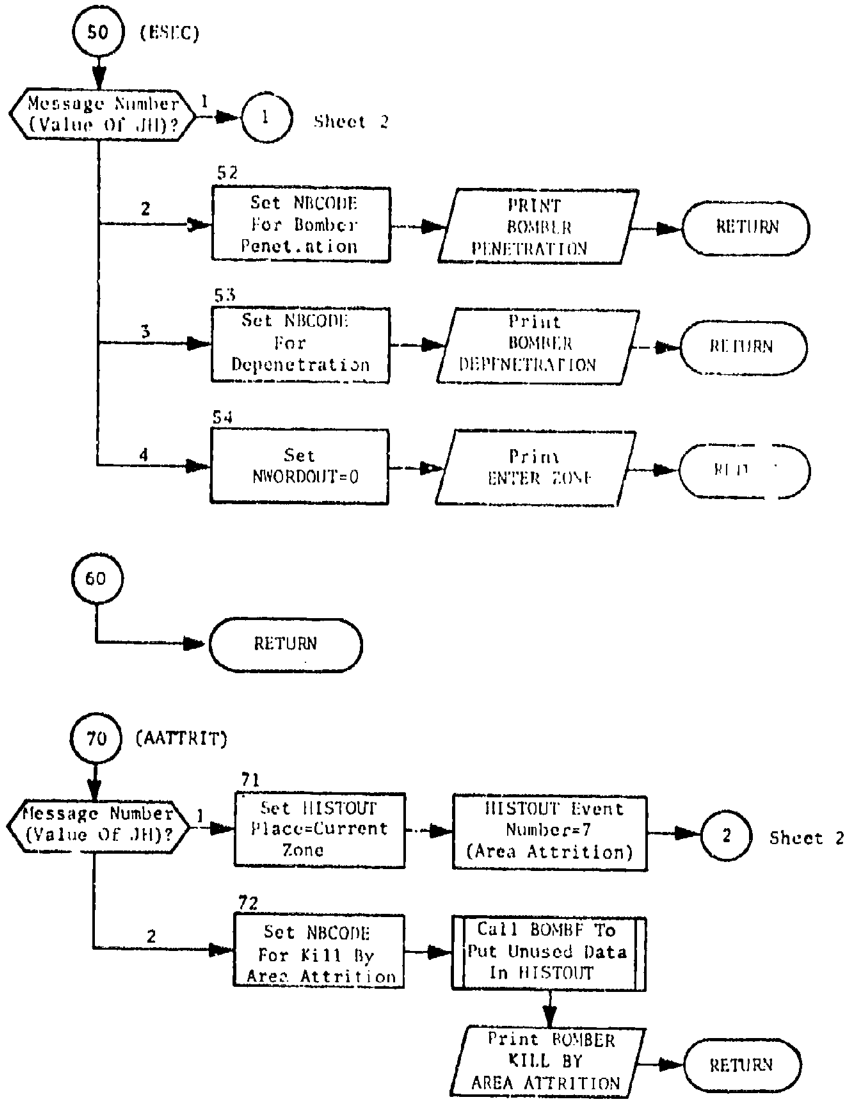


Fig. 25. (cont.)
(Sheet 6 of 13)

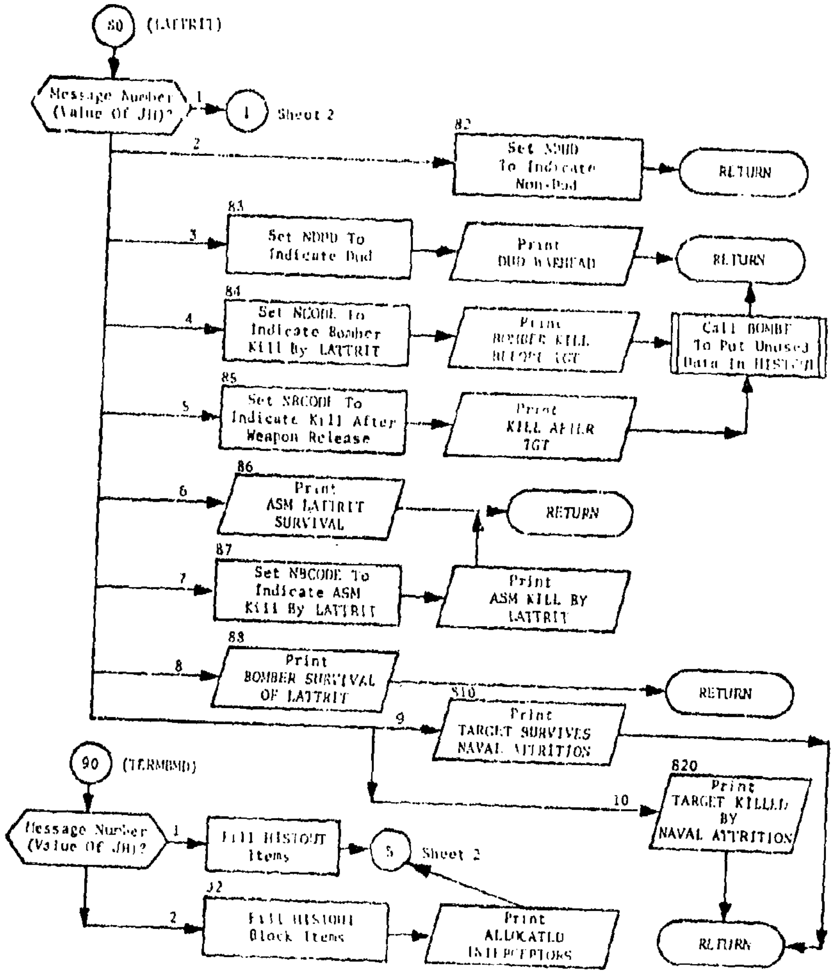


Fig. 25. (cont.)
(Sheet 7 of 13)

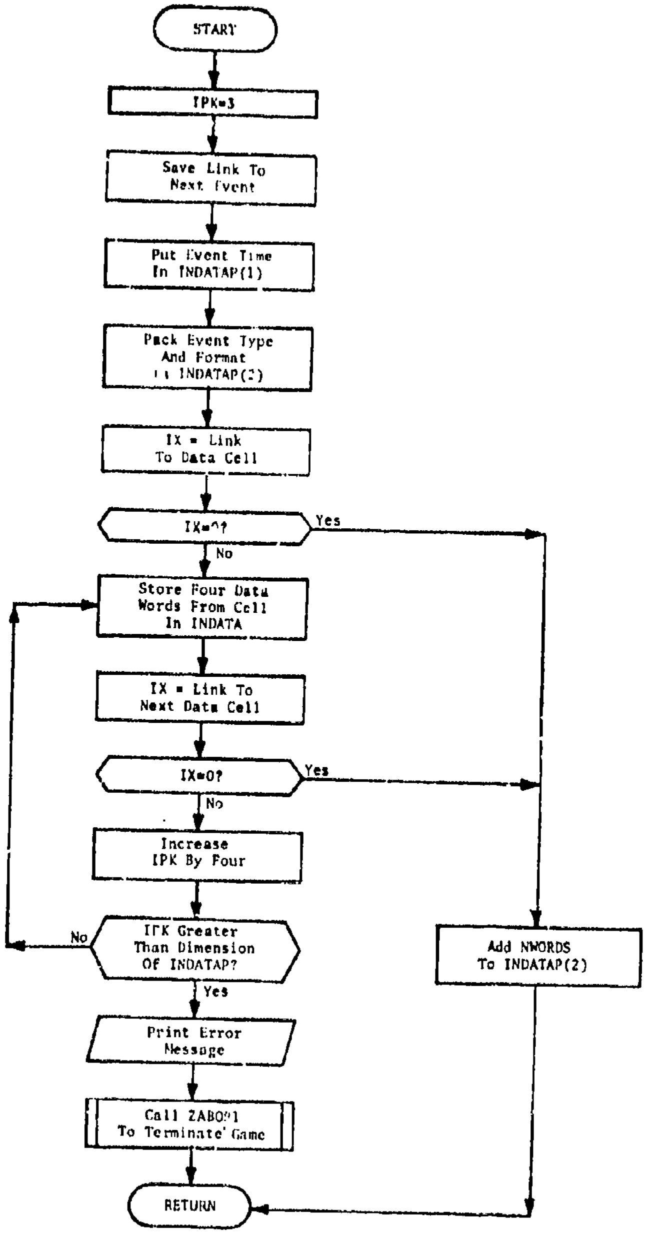


Fig. 51. Subroutine SQUEEZE

SUBROUTINE SSTAT

PURPOSE: To compute and print zone status information.

ENTRY POINTS: SSTAT

FORMAL PARAMETERS: None

COMMON BLOCKS: HISTOUT, IPRINT, NAMES, NWORDOUT, TIME, ZONES, ZSTATUS

SUBROUTINES CALLED: HISTWRIT, PLANTS

CALLED BY: DONEXT, ENDGAME

Method

JT, the number of zone status events executed, is increased by one. The following computations are made for both of the two sides. NPENTOT, the total number of penetrators, is initialized to zero. IBEG is set to MINZONE(K), the minimum zone number for the current side. IEND is set to MAXZONE(K), the maximum zone number for the current side. NPENZ(I), the number of penetrators in each zone I from IBEG through IEND, are added together and stored in NPENTOT.

If NPENTOT is nonzero, current game time TIME and the name of the current side NAMESIDE(K) are printed. For each I, the zones from IBEG through IEND, the following information is printed: I, the zone number; NPENZ(I), the number of penetrators in the zone; ZDEFPOT(I), the defensive effectiveness in the zone; ZCCPOT(I), the command and control effectiveness in the zone; and KILLZ(I), the number killed in the zone by area attrition.

If NPENTOT is zero, no printout is made for the current side.

After the above has been executed for both sides, JT is compared to JTMAX, the desired number of zone status events to be executed. If JT is equal to JTMAX, the subroutine exits. Otherwise FUTIME, the time for the execution of the next zone status event, is set to TIME plus one-quarter hour. JOANNA, the second parameter in the call to subroutine PLANTS, is set so that PLANTS does not reference the format index arrays, which SSTAT does not use. Subroutine PLANTS is called to plant a zone status event for execution at FUTIME. NWORDOUT, the number of words of the HISTOUT block to be used, is set to zero, and the subroutine exits.

If the zone status print (option 17) is not requested, this event is planted to maintain the random number and event sequence. No information is printed, however.

Subroutine SSTAT is illustrated in figure 52.

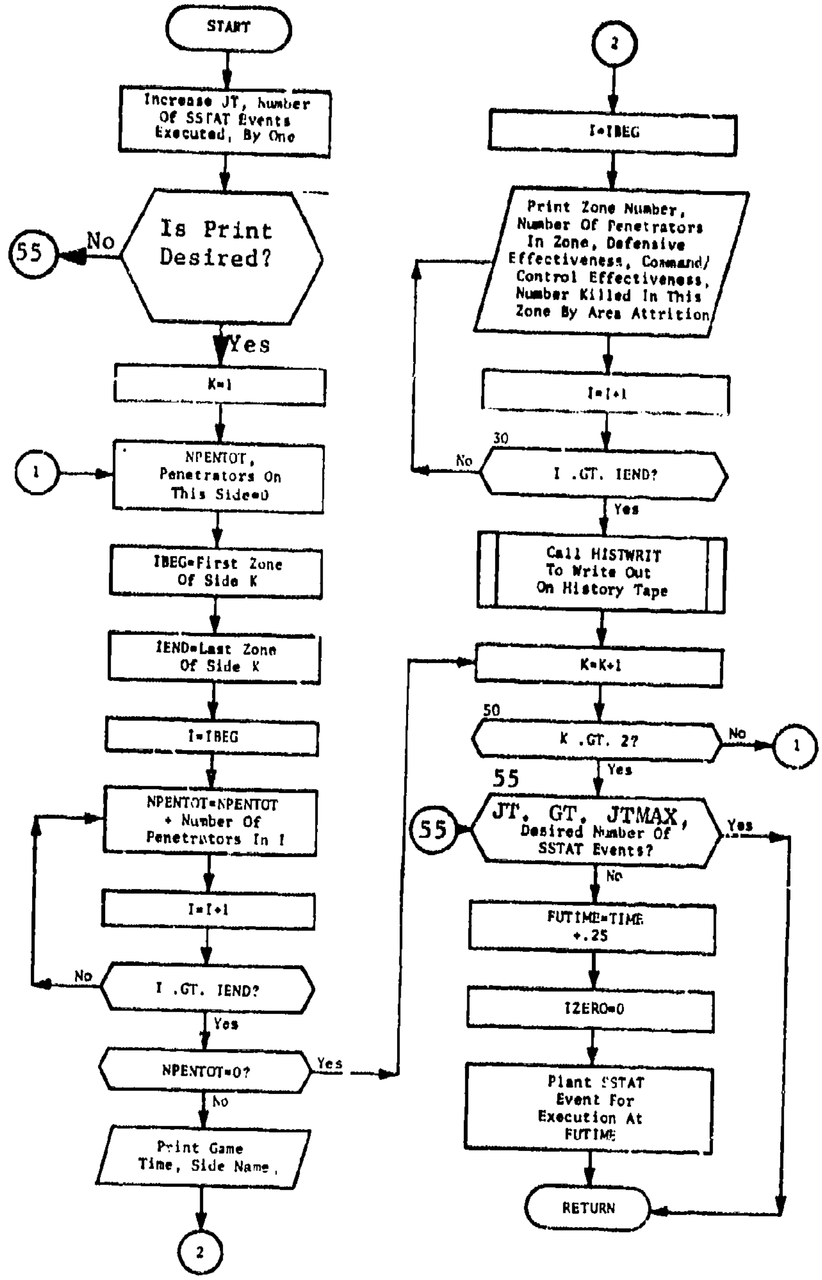


Fig. 52. Subroutine SSTAT

SUBROUTINE STATSUM

PURPOSE:	To print a summary of the status of game items by class and type.
ENTRY POINTS:	STATSUM
FORMAL PARAMETERS:	None
COMMON BLOCKS:	BRKPNT, KEYWORDS, NAMES, 19501
SUBROUTINES CALLED:	IGET, PAGESKP
CALLED BY:	ENDGAME

Method

A page is ejected on the standard output, and NTYPES is set to NTYPECUM(15), the total number of types in all classes.

For each type, the following computations are made. The number of survivors NSURV(I) is initialized to zero. IBEG is set to the first index of the current type and IEND to the last. Since all items were alive initially, NINIT(I), the number of items of the current type alive initially, is found from the two indices. Through successive calls to subroutine IGET, the status indicator in the STATUS array word for each item of the current type is tested. IGET returns a zero for a dead item and a one for a live item, so NSURV(I), the number of items of the current type surviving, is computed by adding together all the statuses.

NT, the number of types, is initialized to zero.

For each class, the following computations are made and results printed. The current class name NAMCLASS(K) is printed. For each side, the following is done. The current side name NAMESIDE(L) is printed. If the current side is side one, BLUE, IBEG is set to the number of types NT plus one. IEND is set to NT plus NBLUETYP(K), the number of Blue types in the current class. If the current side is side two, RED, IBEG is set to IEND plus one and IEND and NT are both set to NTYPECUM(K), the total number of types in the current class.

In either case, using IBEG and IEND as limits, the name of the current type NAMETYPE(I), the beginning base index for the type, and NINIT(I) are printed for all types. In addition, if the type is one for which

ENTRY POINT	TO SUBROUTINE
REORDER	REORDER
SETHEAD	SETHEAD
SETREAD	SETREAD
SETWRIT	FILEHNR
SETWRITE	FILEHNR
SKIP	SKIP
SSKPC	SSKPC
STORAGE	STORAGE
TERMTAP	TERMTAP
TERMTAPE	TERMTAP
TERMTPE	TERMTAP
TIMEDAY	TIMEDAY
TIMEME	TIMEME
WARNING	ABORT
WRARRAY	RDARRAY
WRITEDIR	WRITEDIR
WRITEDR	WRITEDIR
WRWORD	FILEHNR

DISTRIBUTION

Addressee	Copies
NMCSSC Codes	
B121	3
B122 (stock)	6
B200	1
B210	2
B220	29
B600	1
DCA Codes	
920	1
950	1
System Engineering Facility, ATTN: T221 Reston, Virginia 22070	1
OJCS Studies, Analysis and Gaming Agency, ATTN: SFD, Room 1D957, Pentagon, Washington, D.C. 20301	5
Commander-in-Chief, North American Air Defense Command ATTN: NPPG, Ent Air Force Base, Colorado 80912	2
Commander, U.S. Air Force Weapon Laboratory (AFSC) ATTN: AWL, Kirtland Air Force Base, New Mexico 87117	2
Director, Strategic Target Planning Offutt Air Force Base, Nebraska 68113	2
Chief of Naval Operations, ATTN: OP963G Room 5E531, Pentagon, Washington, D.C. 20350	2
Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314	12
	70